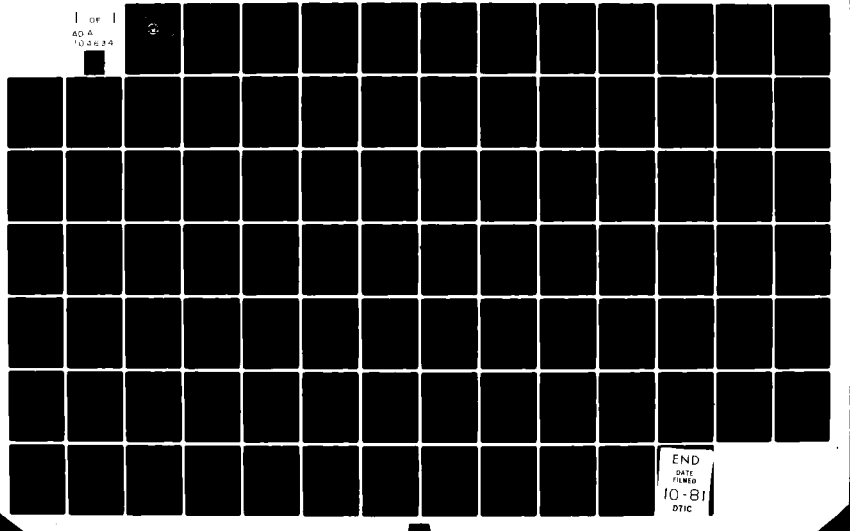
AD-A104 634 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 9/2
NON-DISCRETIONARY SECURITY VALIDATION BY ASSIGNMENT.(U)
JUN 81 L J SHIRLEY

UNCLASSIFIED NL



(2) LEVEL II

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

NON-DISCRETIONARY SECURITY VALIDATION BY ASSIGNMENT.

by

Lawrence Jay Shirley

June 1981

Thesis Advisors: Lyle A. Cox, Jr.
Roger R. Schell

Approved for public release; distribution unlimited.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A104634	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) NON-DISCRETIONARY SECURITY VALIDATION BY ASSIGNMENT	5. TYPE OF REPORT & PERIOD COVERED Master's Thesis June 1981
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Lawrence Jay Shirley	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	12. REPORT DATE June 1981
	13. NUMBER OF PAGES 92
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.	
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)	
18. SUPPLEMENTARY NOTES	
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Computer Security Protection Mechanism Protection Domain Multics Ring Mechanism	
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The assignment technique is a simple mathematical method for determining that a computer protection mechanism is sufficient to enforce specific security policies. The intrinsically inseparable relationship between protection mechanisms and security policies is established.	

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Approved for public release; distribution unlimited.

Non-Discretionary Security Validation by Assignment

by

Lawrence J. Shirley
Lieutenant, United States Coast Guard
B. S., United States Coast Guard Academy, 1973

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1981

Author: Lawrence J. Shirley

Approved by: Tyler A. Colwell
Co-Advisor

Roger R. Schell
Co-Advisor

Chairman, Department of Computer Science

W. M. Woods
Dean of Information and Policy Sciences

ABSTRACT

The assignment technique is a simple mathematical method for determining that a computer protection mechanism is sufficient to enforce specific security policies. The intrinsically inseparable relationship between protection mechanisms and security policies is established.

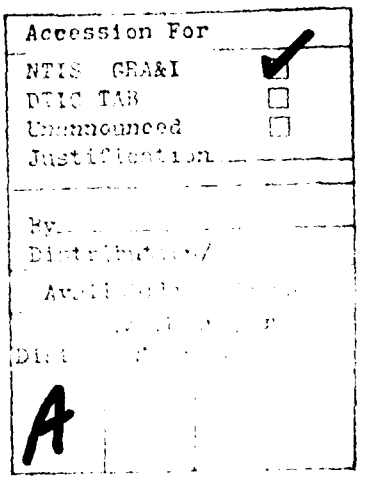


TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

Recognizing the relationship between policies and mechanisms has been a problem in the specification and design of many computer systems. What is needed is a simple methodology for assessing the suitability of a protection mechanism to enforce a non-discretionary security policy. Such a methodology, based upon the entity-relationship model and designed with validation of security enforcement as its primary objective, is presented.

Defined as the assignment technique, this mathematically oriented methodology establishes a relationship between the information sensitivities of the systems entities (partitioned according to the policy constraints), to dominance domains (inherently established by a mechanism). The assignment technique provides a means for mechanism sufficiency validation, since the results of the assignment can be evaluated to determine whether the constraints of the policy are met.

Mechanisms are defined as procedural specifications that prevent the occurrence of operations. Protection mechanisms, then control a subject's access to an object, by adhering to some procedural specification of access rules. Policies, however, are generally stated in a non-procedural form. This

leads to a problem in translating policies into mechanisms, and in verifying the accuracy of this translation.

Only non-discretionary security policies are discussed in detail. Such policies, however, are extremely important when dealing with protection of business information as well as National Security. Computer systems designed to provide Command, Control and Communications must rely upon effective non-discretionary security if they are to be of any value to National Defense [1]. Compromise and subversion policies [2] precisely define the requirements, but the suitability of a protection mechanism to meet these requirements is not always apparent. A theoretical foundation from which this suitability may be simply and readily derived is established.

A. BACKGROUND

Non-discretionary policies for the security of sensitive information have existed throughout the annals of history. The basis of these policies lies in a subject (i.e., an active entity) being prohibited modification or observation of an object (i.e., a repository for information or inactive entity) based upon the subject's membership in a specified group. This grouping is established external to the system in which it will be used.

The first computer systems dealt with the problem of security by establishing physical protection perimeters.

Walls, locks and marines with rifles provided the environment necessary for system security. This was an acceptable procedure because there were relatively few users of the system and each user was trusted not to violate the security policies. Security was an issue external to the computer itself.

However, as computer technology became more sophisticated, user expectations increased. Policy-makers established security policies and expected their machines to adhere to them without exception. The security perimeters that had been established external to the computer, were now to be established internally.

This led to two fields of research. One group, the experimentalists, attempted to design ingeniously contrived mechanisms with little or no concern for the policies which their mechanism would support. Mathematicians, on the other hand, set about the task of modeling policies in a fashion that would establish a foundation for the procedural specification of protection mechanisms. The relationship between these models and the mechanisms was not always clear.

What is needed, and what is presented here, is a simple, complete and consistent means of establishing that a mechanism actually enforces the policy-makers' specifications. This is done by first giving the policy-maker a tool to precisely describe his policy and

then giving the systems designers and analysts a technique to evaluate the sufficiency of their mechanism to support this policy.

A careful examination of the fundamental nature of non-discretionary security policies and protection mechanisms is made. This examination is based largely upon the findings of research associated with security kernel technology [3]. The results of this examination show what it is about mechanisms that actually provides the protection and what protection is actually provided. In so doing, a theoretical mathematical foundation is established from which the science of secure computation may proceed to meet the requirements of the policy-maker in a simple, elegant and efficient manner.

B. RELATED WORK

Research in establishing the suitability of protection mechanisms to meet non-discretionary security policies is practically non-existent. Protection mechanisms are usually presented in an informal manner with implementation details dominating the discussion [4]. Policies, on the other hand, are generated by persons who rarely give consideration to the implementation of these policies in a computer system. The disparity between these two groups has led to little research in methodologies for bridging the broad gap between

security policies and protection mechanisms, and even less results.

The notion of domains originated with Dennis and Van Horn [5] and their concept of spheres of protection. This idea was improved upon by Lampson [6,7] who coined the term "domain" and noted the usefulness of domains as a conceptual tool for understanding protection mechanisms. Schroeder [8] made use of these ideas to design a protection mechanism that would allow mutually suspicious subsystems to cooperate in a single computation.

Popek [9] modeled the nature of access control with his restriction graphs. Bell and LaPadula [10] made a significant contribution when they identified a mathematical framework within which to deal with the problems of secure computer systems. Their work was based upon general systems theory and finite state automata. Furtek [11] established a similar, less known, mathematical framework based upon the theory of constraints. The Bell and LaPadula work was followed by Walters [12] development of a lattice model for security policies. This model was refined and later popularized by Denning [13] such that today, nearly all practical policies have been recognized as lattice policies.

Saltzer and Schroeder [14] presented a tutorial on the basic principles of protection in computer systems. Cohen [15], however, provides a far more rigorous discussion of protection mechanisms while Gronns' [16] research provides

considerable insight into a number of details regarding access relations.

Much of this early work was directed towards the solution of the computer security problem in National Defense [12,17]. As such, the authors rarely discused the motivation for their efforts. It was Schell [1], however, who dramatically described the importance of the computer security in a modern electronic environment. Recognition of the significance of this problem motivated the research reported here.

C. ORGANIZATION

The relationship between security policies and protection mechanisms is not obvious. In order to explore this relationship, one must clarify the meaning of security and protection. Only by methodically examining each and every pertinent principle can one hope to establish a mathematical framework which unifies the security policy issues with the protection mechanisms' design.

The nature of non-discretionary security policies is considered first. The meaning of access relations is explored and commonly known policies are discussed.

Next, a formalized notion of domains is presented. A succinct mathematical definition of a domain is offered. The notion of an (access-mode) domain and dominance domains are

introduced as tools for precisely characterizing protection mechanisms.

Section four discusses the theoretical basis for assignment. The assignment technique is explained and a means for simplifying the the number of assignment schemes needed to establish the insufficiency of a mechanism to support some particular policy is derived.

Section five presents detailed applications of simple assignment showing the usefulness of the assignment technique particularly with respect to mechanism sufficiency validation. Section five dispells much of the mystery that surrounds the ad hoc design of secure computer systems.

Every attempt has been made to provide the reader with a clear understanding of the principles of the assignment technique. Readers are encouraged to question these findings and indeed, the fundamentals upon which they are based. Only in so doing, can one hope to grasp the meaning of the principles presented and the utility of the assignment technique in establishing a foundation for secure computer systems.

II. NON-DISCRETIONARY SECURITY POLICIES

This section provides a detailed examination as to the nature of non-discretionary security policies after first discussing several pertinent concepts concerning policies in general. Some of the issues presented may appear to confuse policy issues with mechanism issues. Hopefully, this confusion will be resolved as the reader obtains a thorough understanding of the inherently isomorphic nature of policies and mechanisms, as substantiated in the ensuing discussion.

A. THE NATURE OF A POLICY

The fundamental nature of a policy has not been clearly established in the Computer Science field. For example, Wulf, Cohen, Jones and others suggest that a policy is a mechanism when discussing HYDRA [18]. Jones subsequently discusses how protection mechanisms can be used to enforce security policies [19]. On the other hand, Cohen defines a policy as a problem in his doctoral dissertation [15] but, enumerates several protection problems associated with one security policy [15]. Such confusion among such a closely related group of computer scientists specializing in operating system security is by no means an isolated situation.

Snyder [20] makes note of this problem stating that capability-based protection systems designers rarely consider the security policies their system may implement. Throughout the computer security literature, one may observe that the nature of a policy and how it relates to the protection issues discussed, is often ignored. Perhaps this is because the nature of security policies themselves, and the suitability of protection mechanisms to meet these policies is not clearly understood. It is the intent of this author to address this problem. In order to do so, one begins by formalizing the notion of a policy.

A policy is a specification of behavior. Such a specification constrains the activities within a system by establishing a distinction between acceptable and unacceptable behavior for some set of classes established by the policy. When dealing with the security issue, the classes (i.e., access classes) are simply labels which the policy uses to distinguish between groups of system entities. So a security policy specifies a set of access classes and identifies the acceptable behavior between them.

Enforcement of policies may be realized in a number of ways. In general, any means of security enforcement internal to the computer, may be considered to be a protection mechanism. As such, implementation details are generally ignored.

The term behavior generally implies that an active entity is dealing with some other entity or entities. So one can distinguish between two types of entities with respect to security policy specifications. One type is those entities whose behavior is being controlled. These are the active entities within the system and are referred to as "subjects". The other type is those with which the subject interacts during execution that are not subjects, but rather are simply repositories of information [12]. These are the passive entities within the system referred to as "objects".

A process is characterized by an address space and an execution point or state of its virtual processor. It is important to note the distinction between processes and subjects as these two terms are often incorrectly considered to be synonyomous. A subject is implemented as a process-domain pair [6,7,8]. One must take care not to confuse these two terms.

Much confusion has been associated with the issue of policy enforcement. A policy may be completely enforced in a system, partially enforced in a system or not enforced at all. Partial enforcement applies only to complex policies for which sub-policies can be formulated and enforced. Partial enforcement does not imply enforcement of a policy only under certain conditions, or at certain times, which is, in fact, no enforcement at all. Partial enforcement

refers to enforcement of a sub-policy within the context of the overall policy.

Policies are not problems [15]. Problems occur only in the implementation of a policy and are used to describe pitfalls in the enforcement of some policy of interest.

Applying some policy to a system makes no changes to that system at the time of application. This means that policies do not initially alter the entities with which they deal. Rather, entities are assigned to an access class according to the policy. If an entity is assigned to an access class such that its attributes require modification, or its relationships are invalid, or the entity itself does not belong within the system, the system is not in compliance with the policy. Action may be taken later to bring the system into compliance, but simply associating the policy with the system, in effect, only labels the system entities.

Recognizing the nature of a policy is important if one is interested in enforcement of policies in computer systems. This is because the logical nature of a computing device dictates a logical specification of policy. Having clearly described the nature of a policy in general, one may now examine security policies.

B. SECURITY POLICIES

Security policies are generally grouped into two broad classes. Non-discretionary security policies (sometimes referred to as mandatory policies), are policies which fix the classification of information sensitivities and establish all permissible access relations (viz., subjects gaining some form of access to objects) according to these information sensitivities. Such a policy is generally considered to externally constrain what access is permissible [3]. Enforcement of a policy requires that the sensitivity of all objects and the authorizations of all subjects be clearly identified.

Discretionary policies, in a sense, provide a finer granularity of access control within the constraints of the non-discretionary policies of the system [3]. Authorization to access information and specification of source information access classes are made outside of the computer environment. A policy is discretionary when a subject with access to an object may exercise its discretion in making that object available to some other subject. As such, the information sensitivity of an object is decided in a discretionary or arbitrary manner. This tends to produce "spaghetti bowl" policies where the information sensitivities of objects is not easy to determine. The sensitivity of objects is constantly changing in an arbitrary manner which may not be readily observable or

controllable. Such policies are not practical when dealing with many of the National Defense issues. Because of their limited utility, discretionary policies are not as interesting as non-discretionary policies nor is their enforcement such a critical issue.

Only non-discretionary security policies are examined in this discussion. It is shown that all non-discretionary security policies can be represented as lattice security policies.

C. LATTICE SECURITY POLICIES

A number of non-discretionary security policies have already been described as lattice policies [12,21]. As such, the precise form of the lattice structure is helpful in understanding the nature of the policy [19].

A universally bounded lattice is a mathematical structure consisting of a finite, partially ordered set for which there exists precisely one least common upper element (i.e., the least upper bound (LUB)) and precisely one greatest common lower element (i.e., the greatest lower bound (GLB)) [22,23]. A partially ordered set, is a set, Q, for which a relation, R, is applied to Q such that R is reflexive, antisymmetric and transitive [22]. For example, consider the set $Q = \{ q_1, q_2, q_3, q_4 \}$ and the relation R applied to Q such that $q_1 R q_2$ (i.e., q_1 is related to q_2 by relation R), $q_1 R q_3$, $q_1 R q_4$, $q_2 R q_4$, and $q_3 R q_4$. The relation R

forms a lattice on the set Q with q_1 as the GLB and q_4 as the LUB.

When discussing lattice security policies, one recognizes the set Q as the set of access classes established by the policy. The access relation R, however, may vary significantly from policy to policy. This fact is not so well recognized. Dennings information flow model [13], for example, describes a flow relation, "$\longrightarrow$", defined on pairs of access classes such that for classes A and B, A $\longrightarrow$ B if and only if information in class A is permitted to flow into class B. This relation applies to compromise and subversion policies, for example, but is meaningless when discussing program integrity.

Three relations between access classes are generally sufficient to describe the specifications of any non-discretionary security policy. For access classes A and B, these are :

A > B	Information of access class A is more sensitive than information of access class B
A = B	Information of access class A is of the same sensitivity as information of access class B
A # B	Information of access class A is in no way related to information of access class B

The notion of sensitivity may be easily confused when discussing several policies. This is because the term takes

its meaning from the policy in question and cannot be readily associated with two diverse policies. For example, an object O may be > a subject S with respect to one policy, # with respect to another policy, and S > O with respect to still another policy. Sensitivity, then, may not be useful for discussing multiple policy issues. It is however, a useful intuitive term for describing the lattice nature of a policy.

This author advances the hypothesis that all non-discretionary security policies may be represented as lattice policies. A simple argument is offered in support of this hypothesis as a complete proof has not been developed.

Non-discretionary security policies are established external to the computer system environment. As such, they define some form of behavior between subjects and objects from which the system may not deviate without external authoritative approval. The system entities (i.e., the subjects and objects) must be clearly labeled or otherwise identified with respect to the policy. Grouping those system entities whose labels are identical, one may establish a set of equivalence classes which completely partition the systems' entities. One may think of these equivalence classes as labeled by the access classes. Such a partitioning, for all practical policies and systems is finite.

One may then examine the relations between access classes with respect to the policies. Enumerating all the relations between access classes, one may draw a graph, such as that shown in figure 1, with nodes signifying access classes and arcs signifying that the access class of the higher node (i.e., closer to the top of the page) is more sensitive (>) than the access class of the lower node. Transitive relations need not be drawn as their inclusion is implicit and does not affect the graph.

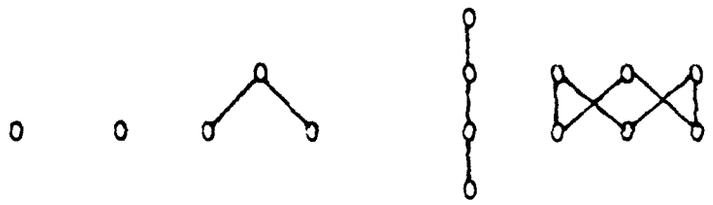


Figure 1. Disjoint Partially Ordered Sets and Nodes

If any cycles are discovered, in an attempt to construct the graph, one may see that the specification of policy is not enforceable. That is to say, for some cycle of access classes A > B > ... > Z > A, the information sensitivity of some access class A is at the same time > A and = A. This is a paradox. Attempting to enforce such a specification is intuitively nonsense! So if one is to have a non-discretionary security policy, viz., one which is to be enforced in a mandatory fashion, one may safely assume that the policy will specify no cyclic relations among the access

classes. Therefore, one may categorically state that the graph of any enforceable non-discretionary security policy will never contain any cycles.

Further examining the graph, one can observe that only two general structures may exist. The first consists of unrelated nodes (i.e., those nodes which are singletons representing access classes with no relations to other access classes in the graph). The other structures are partially ordered sets (some of which may be a lattice).

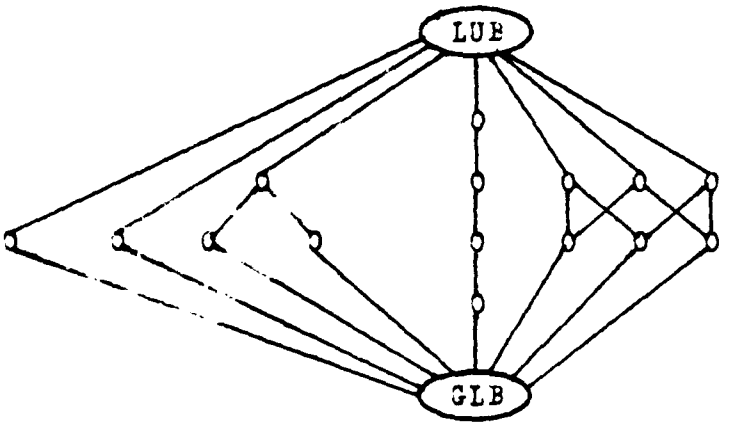


Figure 2. Lattice Structure

If the graph does not contain a least upper bound, (LUB), one may arbitrarily create an access class so designated and establish the appropriate relations with respect to its sensitivity (see figure 2). This access class may also be referred to as the "system high." Likewise, one may do the same for the greatest lower bound (GLB) which is generally known as the "system low." Note that, neither the

LUB nor the GLB need have any entities associated with their access class. By forming this structure, one has established a lattice.

Thus, all non-discretionary security policies are lattice security policies. Non-discretionary security specifications that generate cyclic structures are not well formed policies and as such, their enforcement cannot be evaluated nor can one consider such a specification to be a policy worthy of discussion.

D. SIMPLE LATTICE SECURITY POLICIES

A policy is a "simple lattice policy" when the policy establishes either one of two basic lattice structures. The first structure is formed by a simply ordered (viz., linearly ordered or totally ordered) set of access classes. For example, some policy might establish a simply ordered structure where SECRET is more sensitive than (>) CONFIDENTIAL > UNCLASSIFIED. Policies with simply ordered sets of access classes are called "hierarchical policies."

The other basic lattice structure is formed by a mutually exclusive set of access classes. For example, some policy might establish a mutually exclusive structure where CRYPTO is not related to (#) NATO # NUCLEAR. Those policies with mutually exclusive sets are called "category policies." One should note that, a "compartment" access class, e.g., CRYPTO-NATO, is formed when some restricted form of access

is available to two or more otherwise mutually exclusive categories of information.

Recall that a lattice security policy partitions the systems entities with respect to their information sensitivities, into a set of equivalence classes that can be labeled by the access classes. Consider any two lattice security policies, P_1 and P_2, and some system containing a non-empty set of entities, A. When P_1 is applied to the system, a partition, π_1, is established creating the set of equivalence classes, $\{ e_1, e_2, \ldots, e_i, \ldots, e_n \}$. Applying P_2 to this system so partitioned, refines the system producing a unique partitioning π. π then, is simply the product of π_1, the partition induced by P_1 and π_2, the partition induced by P_2. So for each e_i, an equivalence class created by P_1, a new set of equivalence classes, $\{ e_{i1}, e_{i2}, \ldots, e_{in} \}$, is produced. The partition π forms a lattice, viz., that induced by the composite policy P.

It readily follows that all lattice security policies are the product of one or more simple lattice policies. The total non-discretionary security package for a system then, consists of some set of simple lattice security policies successively refining the systems entities, none of which may produce conflicting policies. This is shown to be particularly useful knowledge when one attempts to use the assignment technique as a means of security validation.

E. ACCESS RELATIONS

Any specific non-discretionary security policy will distinguish one or more distinct access relations between subjects and objects. Associated with these distinctions one may derive, where not otherwise specified, the set of "access rights" which may be accorded to the subject. These access rights specify the liberties which the subjects may take with respect to these objects. Access rights are typically mirrored in the "access modes" of the corresponding protection mechanism. Although there exists a fine difference between an "access right" and an "access mode", viz., "access rights" are associated with security policies and "access modes" are associated with the protection mechanisms which enforce the policy, this discussion frequently refers to an "access right" as an "access mode" because it is the access mode which must inevitably be questioned when evaluating the enforcement of a security policy.

The enforcement of a policy is fundamentally limited by the system's granularity of access which may also be thought of as the system's variety or richness of access modes. Policies that prescribe distinctions not recognized by the access control mechanisms must be enforced in an overly restrictive manner or ignored. For example, a policy addressing a concatenation access relation cannot be

precisely enforced on a system that does not recognize some form of append access mode.

The basis of all security enforcement evaluation lies in the acceptability of an access relation. An access relation is defined as a tuple (subject, access mode, object). This tuple signifies that a relation between the subject and object exist such that the subject is permitted to access the object with all the privileges associated with the access mode. The problem of information security may generally be expressed as the problem of permitting the existence of only those access relations that in no way violate any of the applicable systems policies.

One can see then, that the granularity of access control within a system is dependent upon the ability to distinguish attributes of subjects and objects plus the distinct access modes available. The primitive access modes (i.e., those access modes that are not decomposable by the system) associated with the design of the system, including the protection mechanisms, designate the associated rights accorded to an access request.

When the granularity of access is successively refined, one may observe two conflicting phenomena. First, the ability to distinguish between access relations is more pronounced, thus allowing for greater sophistication and variety in policy formulation. The problem, however, is that the increased distinctions of access relations increases the

complexity of the security evaluation process. Systems designers are faced with the problem of striking a balance between the granularity of access and the complexity of system security validation.

This has not deterred the efforts of many systems designers, however, as the granularity of subjects and objects is quite refined in many systems. Unfortunately, such systems, almost without exception, have failed to enforce even minimal non-discretionary security policies.

Two generic access modes are particularly useful in the discussion of security. These are [16] "observe" (the ability to observe information) and "modify" (the ability to modify information). Other access modes may be generally thought of as a finer granularity of these two access modes. Figure 3 illustrates one such possible set of primitive access modes and how they are associated with the generic access modes.

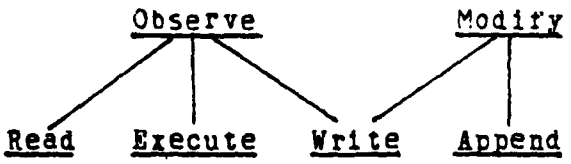


Figure 3. Generic Access Modes

The problem of computer security enforcement can be reduced to the problem of limiting the access relations within the system to only those that neither directly nor

indirectly violate the systems security policies. If one can establish that all of the access relations permitted in the system are acceptable to the policy, one has established that the system is "secure."

F. ILLUSTRATION OF POLICIES

In reviewing the computer science literature, this author was unable to discover any illustration forms appropriate for showing the features of non-discretionary security policies in sufficient detail that one could readily discern all permissible access relations within the system simply by examining the illustration alone. This section presents a review of the major forms examined and their failure to adequately illustrate access relations. It also provides two proposed alternative forms that more clearly illustrate access relations of a system in a manner which leaves no doubt as to the nature of the policy and the requirements for its enforcement.

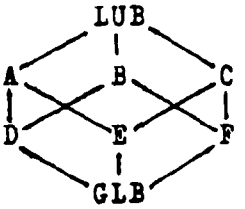


Figure 4. Basic Lattice Form

Figure 4 shows a representation for a lattice structure commonly found in mathematical texts [22,23]. With respect

to lattice security policies, each node represents an access class and the arcs signify that the node nearer the top of the page represents an access class which is more sensitive than the lower nodes' access class. Thus, in figure 4 one may observe that A > D and B # A. Sometimes these arcs are labeled by ">" symbols, but this merely tends to clutter the illustration and provides no additional information. Note that this form provides no information regarding access relations without some examination of the policy that is being illustrated, e.g., one cannot readily answer the question "can a subject of access class A write to an object of access class D?"

The form shown in figure 5 [12,13], provides basically the same information. This form illustrates the permissible information flow that is immediate and non-reflexive by means of directed arcs. Nodes are once again used to represent access classes. Access relations are still non-discernible by examination of the illustration alone.

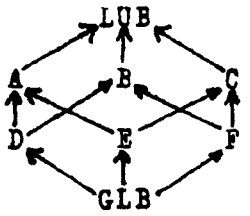


Figure 5. Information Flow Form

Another form which is popular in capability-based protection systems research [24], illustrated in figure 6, is called a protection graph [20]. These graphs specify each subject as a solid node, "●", and each object as an empty node, "○". The directed arcs between nodes specify the access rights of the source by the associated labels. This form provides an extremely detailed means of representing all access relations within the system. Unfortunately, this form provides such detail that an illustration of any practical system becomes exceedingly busy. Thus one quickly loses the ability to distinguish between access classes even when they are clearly labeled. What is needed is needed is a higher order of abstraction for the presentation of practical systems.

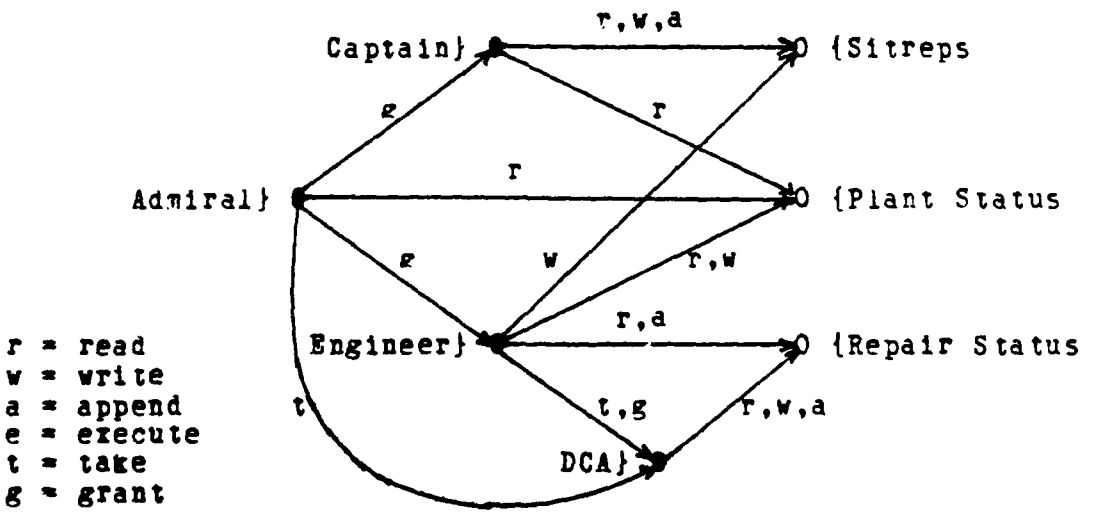


Figure 6. Protection Graphs [20]

Figure 7 represents the first illustration form proposed by this author called an "access relation graph". In this form, each node represents an access class as specified by the policy. All non-reflexive immediate access relations [13] between access classes (except those that may be established by forming a transitive closure over some given access mode(s)) are grouped by access mode and shown as directed arcs labeled by the associated access mode(s). This form solves the problem of the protection graph for non-discretionary security policy representation by providing the minimum information necessary for one to fully grasp all the security implications of the policy from a single illustration.

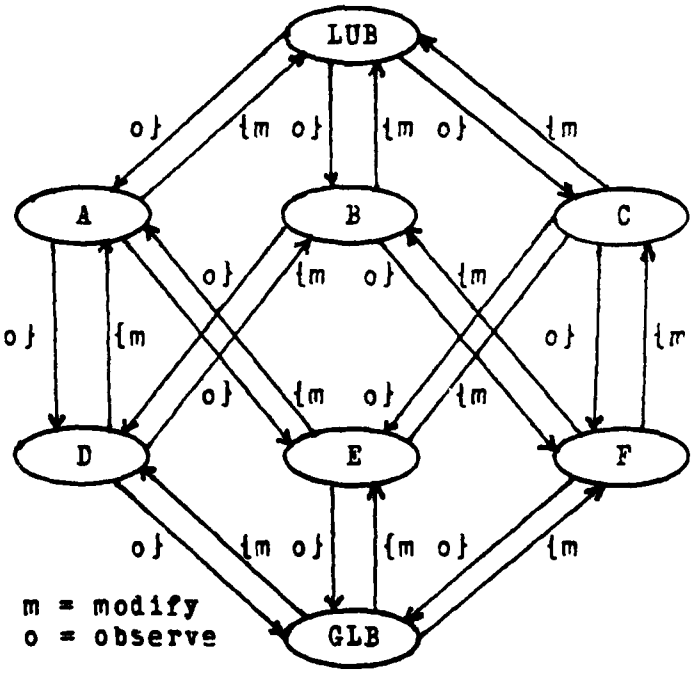


Figure 7. Access Relation Graph

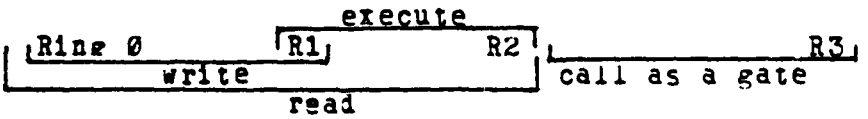
An access relation graph clearly shows all permissible access relations specified by a non-discretionary security policy. Reflexive relations, i.e., those with a subject of the same access class as the object, need never be specifically cited unless all access modes are not permitted within an access class. Antisymmetric relations are clearly defined by the directed arcs. Transitive relations are inferred from the path of two or more antisymmetric relations (viz., in figure 7 a subject of the LUB access class may read from an object of the GLB access class). Therefore, the form meets the mathematical requirements for a lattice in that, all access relations for the lattice (i.e., a universally bounded partially ordered set) are clearly illustrated.

In its most delineated case, the access relation graph is reduced to a protection graph. The advantage of the access relation graph over the protection graph is simplicity. Only the access relations needed to represent the policy are shown. Additionally, complex policies and composite policies are illustrated in one simplified form.

Another illustration form that is particularly useful when discussing uniform lattice structures (i.e., those access relation graphs where the access modes between any two antisymmetric access classes are identical) is the linear access graph. Such a graph shows the security label(s) of the objects (i.e., how one represents the

sensitivity of the object) and denotes the access modes available to subjects of varying sensitivity with respect to the sensitivity of the objects. Figure 8(A) illustrates a simple general linear access graph. In this figure, subjects with greater sensitivity than the objects sensitivity would enjoy the use of access mode(s) 2 when referencing that object. Subjects of inferior sensitivity than the objects sensitivity would enjoy the use of access mode(s) 1 when referencing that object. Subjects of the same sensitivity as the object would enjoy access modes 1 and 2 when referencing the object. The linear access graph for the Multics Ring Brackets, first pointed out to the author by R. Schell, is shown as an example of a familiar policy represented in this form in figure 8(B).

(A)



(B)

Figure 8. Linear Access Graphs

The disadvantage of the linear access graph is that it may only be used for illustration of uniform policies, i.e., those policies where the access relations between any two access classes (one of which is more sensitive than the other) are identical. The succinct nature of this form, however, makes it possible to capture the essence of a class of policies, i.e., those which may be described by the same linear access graph, without going into all the details.

G. EXAMPLE POLICIES

Having discussed the nature of policies in general, one is now prepared to examine several specific policies of interest. Such a discussion logically begins with the two broadest classes of security policies, i.e., compromise and subversion.

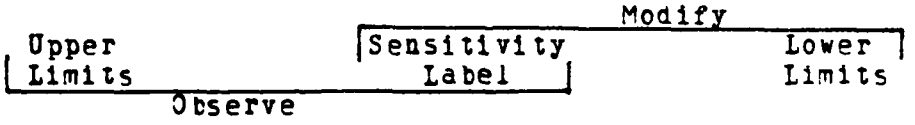


Figure 9. Compromise Policy.

A compromise policy, sometimes referred to simply as a security policy, is one whose primary intent is to prohibit the unauthorized observation of information. Figure 9 show the general form of such a policy. Subjects may observe only those objects whose sensitivity is less than or equal to the subject's sensitivity in order to prevent direct observation

of an object by an unauthorized subject, viz., the Simple Security Condition [10]. In order to prevent indirect observation of objects by unauthorized subjects, a sufficient but not necessary condition establishes that modification of objects must at least be limited to those subjects whose sensitivity is less than or equal to the objects sensitivity, viz., the (Security) Confinement Property -- also known by a less descriptive title as the *-Property [10].

A subversion policy, sometimes referred to simply as an integrity policy, is the dual of a compromise policy. The primary interest of a subversion policy is to prohibit the unauthorized modification of information. Figure 10 illustrates these general characteristics. Subjects may modify only those objects whose sensitivity is less than or equal to the subject's sensitivity in order to prevent direct modification of an object by an unauthorized subject, viz., the Simple Integrity Condition [21]. In order to prevent indirect modification of objects by unauthorized subjects, a sufficient but not necessary condition is that observation of objects must be limited to those subjects whose sensitivity is less than or equal to the object's sensitivity, viz., the Integrity Confinement Property [21].

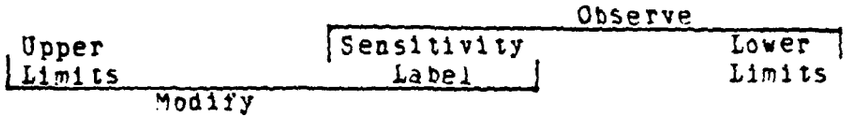


Figure 10. Subversion Policy.

The importance of subversion policies should not be underestimated [2,21]. Changing the course of an ICBM, for example, should in most cases require a more sensitive authorization than simply knowing its course. Such policies, however, are often overlooked in many Command, Control, and Communications systems [2].

Another general class of policies that is of general interest in Security Kernel research, and whose title was coined during the course of this research effort by R. Schell, are the "Program Integrity" policies [4]. The notion of program integrity stems from the desire to prohibit unauthorized modification of executable programs by less trustworthy subjects. In the general case, one wishes to ensure that the more sensitive programs are "tamperproof." In other words, one wants to be sure that the program can be "trusted" to perform as specified and can not be "tricked" by merely reading data of lower sensitivity or "importance." For example, a system designer/programmer may wish to insure that his programs always perform as specified in both his test environment and in any application environment. Unlike a strict integrity policy [21], program integrity is not

concerned with the issue of general observation of information. Program integrity is therefore less conservative (and thus more "risky") than Bibas integrity policy. Program integrity deals only with execution and modification of information. As such, figure 11 illustrates the general form of a program integrity policy.

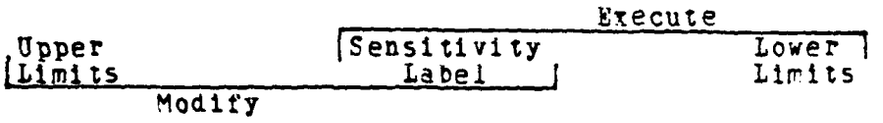


Figure 11. Program Integrity Policy.

One may guarantee that no direct modification of a program by an unauthorized subject (i.e., a direct threat) is possible by enforcement of the following condition :

> Simple Program Integrity Condition : If a subject has modify access to an object, then the program integrity of the subject is greater than or equal to the program integrity of the object.

Because program integrity policies are concerned with the execution issue (versus the observation issue [21]), indirect modification of information is not strictly prohibited. This provides a certain degree of flexibility, but also produces a certain amount of risk [19]. Confinement of execution reduces the risk of such an indirect threat but does not eliminate it. A more sensitive subject must be trusted not to modify a less sensitive object either

intentionally or otherwise. An indirect threat occurs when a subject executes a program that has been modified by a less trustworthy subject, therefore, one wishes to confine the execution access relations. The confinement property for program integrity is defined as follows :

> Program Integrity Confinement Property : If a subject has execute access to an object, then the program integrity of the object is greater than or equal to the program integrity of the subject.

The remainder of the section discusses three policies of general interest to federal ADP users. Any computer system designed for use by the federal government, should as a minimum, consider its ability to enforce these policies.

1. National Security Policy

The National Security Policy classifies information essential to the National Defense or foreign relations of the United States. The President of the United States established this policy in Executive Order Number 12065 dated June 28, 1978 [25]. This order defines three levels of classification as follows :

> TOP SECRET : That information or material the unauthorized disclosure of which could reasonably be expected to cause exceptionally grave damage to the national security.
>
> SECRET : That information or material the unauthorized disclosure of which could reasonably be expected to cause serious damage to the national security.

CONFIDENTIAL : That information or material the unauthorized disclosure of which could reasonably be expected to cause damage to the national security.

Implicit in this set of definitions, there also exists a classification of information which is not classified. Therefore, one has four hierarchical access classes established by this policy, the intent of which is to prevent unauthorized disclosure (viz., observation) of information so classified. Figure 12 shows the access relation graph for this compromise policy which is referred to as the basic National Security Policy.

Executive Order 12065 also establishes [25] the authority to originally classify new information. Information may be classified Top Secret only by officials designated in writing. Information may be classified Secret only by officials who have Top Secret classifications or by officials designated in writing. Information may be classified Confidential only by officials with Top Secret or Secret classifications or by officials designated in writing.

In order to obtain access to classified material, the order indicates that a person must be determined trustworthy (granted clearance) and that access is necessary in the performance of that persons' duties ("need to know"). This is a discretionary policy, however, and will be discussed no further. All classified material shall be

appropriately and conspicuously marked to put all persons on clear notice that the information is classified. Classified material no longer needed shall be promptly destroyed.

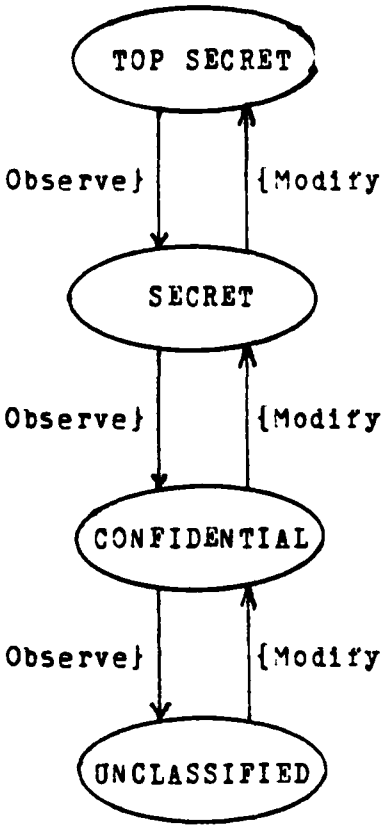


Figure 12. Basic National Security Policy.

2. National Integrity Policy

The dual of the National Security Policy is the National Integrity Policy [21]. Motivation for such a policy comes from the desire to prohibit subversion, i.e., the unauthorized modification of information. The following set of integrity classes have been established for this policy

[21]. Implicit with this classification scheme, one also has information that is not classified.

> TOP SECRET : That information or material the unauthorized modification of which could reasonably be expected to cause exceptionally grave damage to the national security.
>
> SECRET : That information or material the unauthorized modification of which could reasonably be expected to cause serious damage to the national security.
>
> CONFIDENTIAL : That information or material the unauthorized modification of which could reasonably be expected to cause damage to the national security.

One further point concerning Integrity Policies must be emphasized before one proceeds. Generally speaking, one has a good notion of how to classify information with respect to security and unauthorized observation, but classification with respect to integrity is not so easily identified. In some sense, integrity classification must be determined by the object's potential importance rather than by its current importance. Consider, for example, a simple sine function tucked away in some obscure user library. If this function is used to compute trajectories for an inter-continental ballistic missile, it becomes TOP SECRET with respect to the National Integrity Policy, whereas, it is clearly UNCLASSIFIED with respect to the National Security Policy. Classification of information with respect

to integrity will generally require considerable planning and foresight [2].

3. Privacy

The Code of Fair Information Practices and the Privacy Act of 1974 established the following basic policy for the Federal Government [26].

> (1) There must be no personal data record-keeping systems whose very existence is secret.
>
> (2) There must be a way for an individual to find out what information about him is on record and how it is used.
>
> (3) There must be a way for an individual to correct or ammend a record of identifiable information about him.
>
> (4) There must be a way for an individual to prevent information about him that obtained for one purpose, from being used or made available for other purposes without his consent.
>
> (5) Any organization creating, maintaining, using or disseminating records of identifiable personal data must guarantee the reliability of the data for their intended use and must take precautions to prevent misuse.

All information systems (including computer systems) used by the Federal Government are subject to these privacy requirements and must incorporate a corresponding set of safeguards when the process "Privacy Information."

These three policies are applicable to many Federal data processing applications. Numerous other non-discretionary policies exist both in the Federal, State, and Local governments and in private industry. It has been

shown in this section that these policies may be precisely described using access relation graphs or linear access graphs as described in this section. Once a policy has been so described, a precise evaluation of its enforcement may be considered.

III. A FORMALIZED NOTION OF DOMAINS

The notion of a "domain" has not been clearly presented in a precise manner, nor properly defined. Dennis [5] introduced the concept by describing a "sphere of protection." Lampson [6] refined the concept, coining the term "domain", and defined a domain as a group of capabilities or protected names. Schroeder [8] maintains Lampson's definition, but provides an in-depth discussion and presentation of his ideas, many of which were instrumental in the formulation of the concepts presented here. Schroeder further refined the ideas from his thesis, and together with Saltzer [14], defines a domain as a set of objects that may be accessed by a principal. This definition is the most commonly accepted today, but for any rigorous discussion of domains, or for presentation of a concept such as the assignment technique, a more formalized definition is needed.

An access domain Δ, is a tuple, $(a_1, a_2, \ldots, a_i, \ldots, a_n)$, where n is the number of primitive (non-decomposable) access modes in the system and a_i is the set of all objects, $\{ O_1, O_2, \ldots, O_j, \ldots, O_m \}$, accessible by the "i"th access mode. An (access mode)-domain is the set of objects that a process executing in that domain (i.e., a subject)

has the right, or privilege of, accessing according to the rules for that particular access mode.

Consider the following examples of domains:

Δ_1: (Observe(O):{A}, Modify(M):{B})

Δ_2: (O:{A,B,C}, M:{A,B,C})

Δ_3: (O:{A,C,D}, M:{$\emptyset$})

Δ_4: (O:{A,B,C,D}, M:{A,B,C,D})

The observe-domain of Δ_1 (denoted as $O\Delta_1$) is object A and the modify-domain $M\Delta_1$ is object B. Note that simply referring to Δ_1 as containing objects A and B would not provide much insight into the true nature of this domain [14].

The notion of "dominance" with respect to domains was introduced by Grohn [16]. These notions are refined from security dominance and integrity dominance to a more general definition of dominance.

A domain, Δ_i dominates ($\propto$) Δ_j if and only if (iff) for each access mode "a", $a\Delta_j \subseteq a\Delta_i$. This is particularly useful when discussing the relationship between domains with respect to access modes. One can say that for some a_k, $a_k\Delta_i \propto a_k\Delta_j$ iff $a_k\Delta_j \subseteq a_k\Delta_i$.

Continuing with the previous group of example domains, $O\Delta_4 \propto O\Delta_3$, $O\Delta_3 \propto O\Delta_1$, $M\Delta_4 \propto M\Delta_3$, $M\Delta_1 \propto M\Delta_3$, Δ_4

$\propto \Delta_3$ but Δ_3 does not dominate Δ_1. Similar examples can be formulated by the reader.

Dominance domains may be labeled for convenience. In the Multics system, for example, the dominance domains established by the ring mechanism were known as rings and were labeled by ring numbers. Schroeder's protection mechanism also uses numbers as labels for dominance domains [8].

The systems protection mechanisms establish a set of dominance domains that can be used for evaluating the protection mechanisms. These dominance domains dominate all domains that currently exist or may exist within the system. If one can establish the set of dominance domains for the system and one can show that the policy holds for these domains, then one can show that the policy holds for all domains.

A mechanism, in the most general sense, is something that prevents the occurrence of certain sequences of operations [15]. A protection mechanism, or an access control mechanism, can be defined as something that prevents the unauthorized access of information. In the broadest sense, one may include as protection mechanisms such things as walls, patrol dogs and cypher locks. More specifically, though, a protection mechanism for a computer operating system is a procedure, implemented in software, firmware (if there is such a thing) or hardware, that prohibits the

access of objects within a system such that the domain of any process is dominated by some particular dominance domain inherently established by the protection mechanisms.

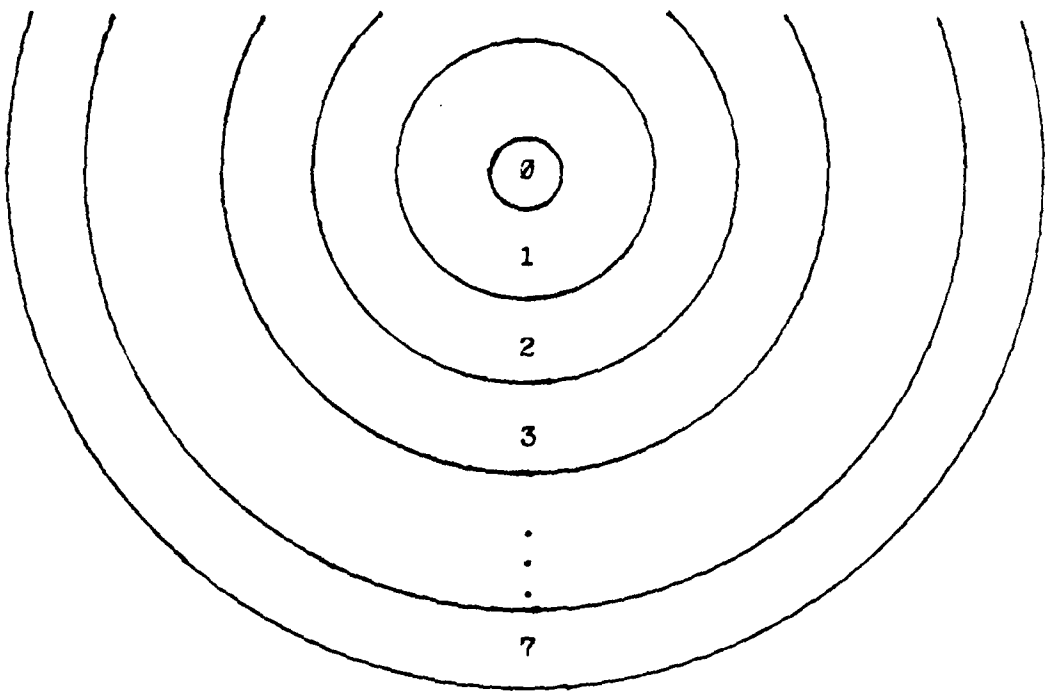


Figure 13. Multics Rings

The Multics Ring Mechanism [28] is a well known protection mechanism that provides an excellent example for the discussion of dominance domains. One may think of these dominance domains as a set of concentric rings (illustrated in figure 13), each numbered in increasing order from the inner-most ring or kernel. The kernel is conventionally assigned ring number zero.

The Multics Ring Mechanism determines the authorized access of a subject by means of the current ring number (r) that specifies the dominance domain. Discrimination among objects is by means of a ring bracket. The ring bracket is a three-tuple (R1, R2, R3) where R1, R2, and R3 are ring numbers and R1 must be numerically less than or equal to R2 which is less than or equal to R3. Access is characterized by the rules illustrated in the linear access graph shown in figure 14.

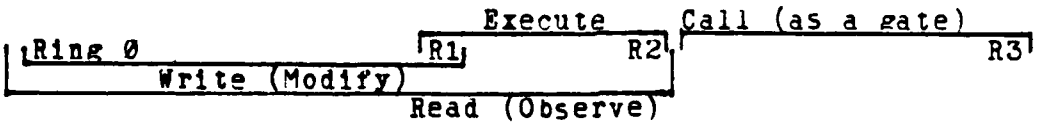


Figure 14. Multics Ring Mechanism Linear Access Graph

Consider now a system that uses the Multics Ring Mechanism and discriminates among four distinct hierarchical rings (0 thru 3). One may think of the domains established by this system as Δ_0, Δ_1, Δ_2, and Δ_3. Consider the rules of access established in figure 14, where $M\Delta_0$ is the objects that may be modified by a process in domain 0. Then $M\Delta_0 \propto M\Delta_1 \propto M\Delta_2 \propto M\Delta_3$. Likewise, $O\Delta_0 \propto O\Delta_1 \propto O\Delta_2 \propto O\Delta_3$. No such relationship exists for execute or call (as a gate). $E\Delta_3$ does not $\propto E\Delta_2$, as R2 may be 2 for some object X, in which case $X \in E\Delta_2$ but $X \notin E\Delta_3$. Likewise $C\Delta_3$ (the Call (as a gate) domain of Δ_3) does not $\propto C\Delta_2$ as R3 may be zero, for example, in which case, R1

and R2 must be zero, ruling out the possibility of successive dominance call-domains.

Note that a single object may be a member of several dominance domains. Some object X, with ring brackets (2,2,3), is a member of $O\Delta_0$, $O\Delta_1$, $O\Delta_2$, $M\Delta_0$, $E\Delta_0$, $E\Delta_1$, $E\Delta_2$, and $C\Delta_3$. Therefore, $X \in \Delta_0$, Δ_1, Δ_2 and Δ_3. This concept can be confusing as an object is a distinct entity generally represented by a single image.

This section has established a formal definition of domains suitable for discussion of complex domain related issues. The notion of dominance domains was introduced and their relationship to protection mechanisms established. The Multics Ring Mechanism provided an example of the means by which one may evaluate the dominance domains established by a protection mechanism. Having formulaized these concepts, the relationship between policy and mechanism may now be investigated in an organized manner.

IV. THE ASSIGNMENT TECHNIQUE

This section introduces a mathematical framework for evaluating the relationship between non-discretionary security policies and protection mechanisms. An evaluation approach, termed "The Assignment Technique", utilizes the entity - relationship model in establishing an assignment between the security classes of information established by the policy constraints, and dominance domains, established by the properties of the mechanism. The assignment technique provides a theoretical foundation for assessing the sufficiency of an access control mechanism with respect to a well formed protection policy.

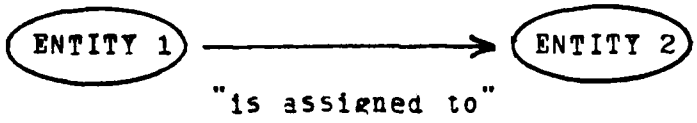
This section begins with a general discussion of the meaning of "assignment". It then proceeds to introduce the assignment technique in a general form. The section concludes with a simplification of the assignment technique made possible by the lattice nature of non-discretionary security policies.

A. ASSIGNMENT

Assignment is the establishment of a relationship between two entities such that the first entity is "assigned to" the second entity. Mathematically, the term assignment is not significant. One could easily have said that entity 1

is related to entity 2. Intuitively, however, assignment is associated with the connotation "to fix authoritatively". This precisely describes the manner in which this relationship is established.

Assignment may be denoted by a graph from the first entity to the second as follows:



It is important to recognize that assignment does not alter either entity. Assignment is merely the act of associating an entity or set of entities with some other entity or set of entities.

Another way to describe assignment is in terms of the act of forming a tuple (entity 1, entity 2). Additionally, one may think of assignment as a function (i.e., "is assigned to") where the assignment process establishes a mapping between two otherwise disjoint entities. Regardless of the context of discussion or the symbolism used, one may simply think of assignment as the act of associating one thing with another.

B. THE TECHNIQUE

The essence of the assignment technique is relatively simple. First of all, consider the nature of a lattice security policy. Such a policy partitions the objects of a

system into a lattice of equivalence classes labeled by the access classes as discussed in section II. Each equivalence class can be thought of as an entity that may be subject to assignment.

Then consider a mechanism, which establishes a lattice of dominance domains as discussed in section III. Each of these domains can also be thought of as an entity that may be subject to assignment.

Since an assignment can be established between any two entities, one can make an assignment between the equivalence classes established by a lattice security policy and the dominance domains established by some protection mechanism. One may then validate that (for this assignment) the mechanism is sufficient to support this policy. This validation is made by examining the set of access relations that the mechanism permits, and testing for possible violations of the policy.

The assignment technique can be described more systematically as follows:

1) Determine if the policy is a lattice policy. If not, the assignment technique does not apply.

2) Establish the set of equivalence classes, $\{ e_1, e_2, \ldots, e_k, \ldots, e_p \}$, that are associated with each access class.

3) Determine the set of dominance domains, $\{ \Delta_1, \Delta_2, \ldots, \Delta_q, \ldots, \Delta_q \}$, that are established by the systems protection mechanism.

4) Make an assignment from e_k to Δ_g.

5) For this assignment, examine the access relations permitted by the mechanism, testing for possible violations of the policy.

6) If no violations can exist, the mechanism is sufficient for the policy in question.

Step 4 of the assignment method allows for considerable flexibility in the manner in which assignments can be made. Any possible mapping from equivalence classes to dominance domains may be considered. This flexibility, however, implies considerable effort in order to determine that a mechanism is not sufficient for a given policy. Fortunately, in this thesis one is specifically dealing with the security issue. Because of this, several refinements can be made that greatly simplify this task.

C. SIMPLE ASSIGNMENT

The question of how one chooses to make assignments (i.e., the choice of an assignment scheme) may seem relatively complex upon first inspection of the assignment technique. The problem, however, becomes almost trivial when dealing with simple non-discretionary security policies as is shown by the following arguments.

First of all, it is clear that the equivalence classes (established by the policy constraints) represent distinct access classes. It is also clear that the dominance domains represent distinct sets of objects. If more than one

equivalence class were assigned to the same dominance domain, then there is nothing in the mechanism to distinguish between the access classes. But the policy does draw some distinctions between these access classes (i.e., that distinction established by the definition of the access classes), so it would not be possible to enforce the policy with such an assignment. All such assignments can be eliminated, a priori.

On the other hand, if one equivalence class was assigned to more than one dominance domain, then some distinction is being made for an access class that is not specified in the policy. In some cases, one may find that such distinctions produce violations of the policy. Although other cases may not do so, these extra dominance domains are unnecessary, providing distinctions which have no significance. Therefore, the number of dominance domains of interest established by the mechanisms should be equal to the number of access classes established by the policies.

One may attempt to argue that there may exist dominance domains that do not receive an assignment. Such domains, however, must be either empty or in no way allow for an exception to the enforcement of the policy. As such, one need not be concerned with the question of their existence. One need only concentrate on the dominance domains for which the assignment was made.

Considering assignment as a function, it has been established that the only assignment schemes of interest are bijective (i.e., a one to one and onto relationship between the access classes and the dominance domains [22]). This provides some improvement, but one is still faced with at least p! possible assignment schemes to evaluate (where p is the number of access classes established by the policy).

One may gain considerable improvement, however, by only attempting to validate one simple mechanism with respect to one simple policy at a time. Furthermore, the knowledge of partially ordered sets may be used to make our assignments in a very selective manner. This is done by first requiring that the lattice for the dominance domains of interest that one considers for assignment, be an isomorphic image of that for the equivalence classes. This may not be a necessary condition, however, it in no way invalidates the results shown (as one would otherwise be dealing with an isomorphic sub-image established by the mechanism), and it is helpful in this discussion.

When considering the isomorphic image of a lattice, the problem of assignment is reduced to a question of orientation. One may either assign the greatest lower bound of the lattice to the greatest lower bound of the image, or assign the greatest lower bound of the lattice to the least upper bound of the image. Any other assignment would not be

acceptable as it would violate the ordering of the lattice or of the image.

So for a system of "k" isomorphic images of the lattice established by the policy, one need only consider at most, 2k assignment schemes. In most practical cases, when the mechanism establishes isomorphic images which are identical in their access control properties because of the uniform nature of the mechanism, one need consider only 2 assignment schemes.

> The Simple Assignment Theorem : For any simple lattice policy and an isomorphic image established by some protection mechanism, no more than two assignment schemes are necessary to validate the sufficiency of the mechanism to enforce the policy.

Proof Sketch : The proof proceeds by showing that two assignment schemes are reasonable and that all others are not.

1) Make assignments starting from the greatest lower bound (GLB) of the lattice to the GLB of the isomorphic image. Then assign every reachable access class (i.e., those of unit distance) to a reachable dominance domain in the isomorphic image. Next assign all reachable access classes from those just assigned (which are not already assigned) to a corresponding reachable dominance domain. Proceed in this fashion until all access classes have been assigned. An assignment such as

that shown in figure 15 will result, where the LUB is assigned to the LUB, A is assigned to A', B is assigned to B', and so forth.

This assignment is a valid assignment in that an assignment can be made from the access classes to the dominance domains that is not inherently incorrect and therefore is worthy of consideration. This does not mean that the protection mechanism is sufficient for this assignment. It only implies that such an assignment scheme is worthy of consideration.

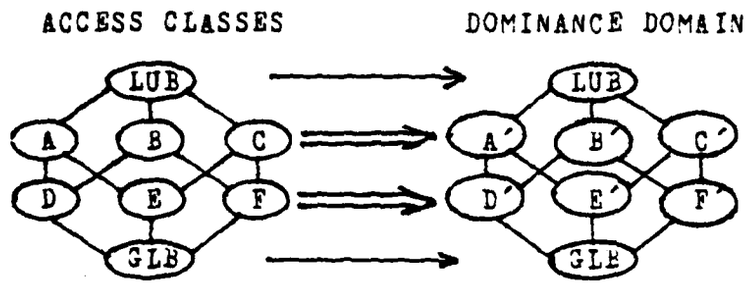


Figure 15. GLB to GLB Assignment

2) Now consider a second practical assignment. This assignment starts from the GLB of the lattice making an assignment to the LUB of the isomorphic image and proceeding as in the first assignment scheme. The resulting assignment is illustrated in figure 16 where the LUB is assigned to the GLB, A is assigned to D', D is assigned to A', and so forth.

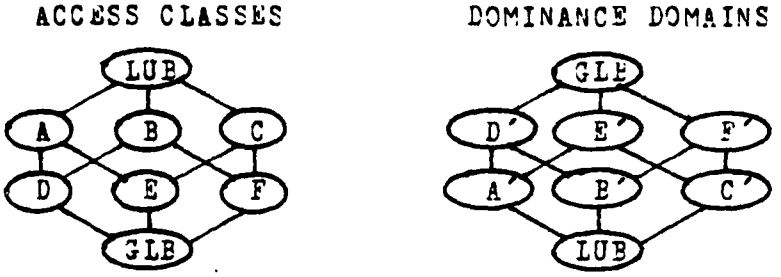


Figure 16. GLB to LUB Assignment.

It is important to note that if the lattice structure is not uniform, i.e., inverting the lattice would not produce the same image, then only one of the two aforementioned assignment schemes will be successful. This limitation occurs because one encounters some set of reachable access classes during assignment that have no corresponding reachable dominance domains. However, for any lattice structure, uniform or otherwise, there will always be one assignment scheme to an isomorphic image that is worthy of consideration. This leads us to the following corollary.

> Corollary 1. For any lattice policy and an isomorphic image established by some protection mechanism, there exists at least one valid assignment scheme.
>
> Proof Sketch (Corollary 1) : The proof is trivial from the definition of an isomorphic image. If a lattice has an isomorphic image, then at least one ordering of nodes in the image is identical to the ordering of nodes in

the lattice, therefore, this ordering is worthy of consideration.

3) Now consider the assignment of the GLB access class to any dominance domain other than the LUB or the GLB. If this is done, then some other access class must be assigned to the LUB dominance domain and still another access class must be assigned to the GLB dominance domain. But if the isomorphic image is to maintain the ordering of the access classes, then there exists some ordering which is not valid because either the GLB or the LUB of the isomorphic image is to be considered less than the GLB (in the image) which must be the least element (viz., least sensitive) according to the policy. Therefore, such an assignment can never be valid. Thus one is reduced to the task of considering only two possible assignment schemes of interest.

One can further simplify the assignment technique by combining steps 4 and 5. This is accomplished by making an assignment and examining all access relations producible immediantly. If an access relation is not valid, one can quickly determine that the assignment scheme in use will not validate the sufficiency of the mechanism.

When one is dealing with more complex lattice structures, one is faced with two alternatives. One can

either validate the sufficiency of the mechanism for each sub-policy, establishing that if each sub-policy is enforced, then the complex policy is enforced, or one may choose to validate the complex policy by a straight forward assignment. When using a straight forward assignment approach, one must remember that the Simple Assignment Theorem may not apply. This is of no particular consequence when validating a protection mechanism designed for a particular policy where the assignments are chosen carefully. However, establishing the insufficiency of an arbitrary mechanism may require considerably more effort.

The basic principles associated with the assignment technique have been presented in this section. One may now consider some simple examples that illustrate the usefulness of assignment.

V. MECHANISM SUFFICIENCY VALIDATION BY ASSIGNMENT

One of the most practical uses for the assignment technique is sufficiency validation of protection mechanisms (i.e., validation of their ability to enforce security policies) [4]. In contrast to other validation techniques [11,17], the assignment technique presents a method whose mathematical model (i.e., the entity-relationship model) is based upon the nature of security itself, rather than other methods which adapt the nature of security into a form designed to mesh with the prescribed format of some well known mathematical model. This section discusses mechanism sufficiency validation by assignment for several well known linear non-discretionary security policies. Although the principles discussed in this section apply for all lattice security policies, only linear lattice policies are discussed in this section as they provide a sufficient foundation for the discussion of any lattice policy and are more clearly illustrated in this context.

A. MULTICS RING MECHANISM ASSIGNMENTS

The question of the sufficiency of the Multics Ring Mechanism for enforcement of the basic National Security policy was the initial problem that prompted the current research effort and led to the formulation of the assignment

technique. It is appropriate then, that this analysis be presented as an introductory application of simple assignment.

1. Compromise Policy

As stated previously in section II, the basic National Security policy is a simple lattice security policy. Figure 13 illustrates this policy.

The dominance domains of the Multics Ring Mechanism are most frequently shown as concentric rings numbered in increasing integer order from the innermost ring or the kernel. The security kernel is generally assigned ring number 0. For simplicity, only a system with rings 0 thru 3 is shown in this analysis. Assignment to other ring numbers (such as 2 thru 5 or 4 thru 7) will produce similar results because of the uniform nature of the Multics Ring Mechanism.

Consider as the first assignment scheme, the assignment of the TOP SECRET access class (the least upper bound of the policy) to ring 0 (the least upper bound of the dominance domains). The assignment produced is illustrated in figure 17.

Next, according the assignment technique, one must examine the access relations permitted by the mechanism and test for possible violations of the policy. In order to do so, one must first examine the nature of the Multics Ring Mechanism more closely. A detailed discussion is given by Schroeder [27], however, a simple explanation of the

pertinent details as used in this discussion is provided for those readers not otherwise familiar with Multics.

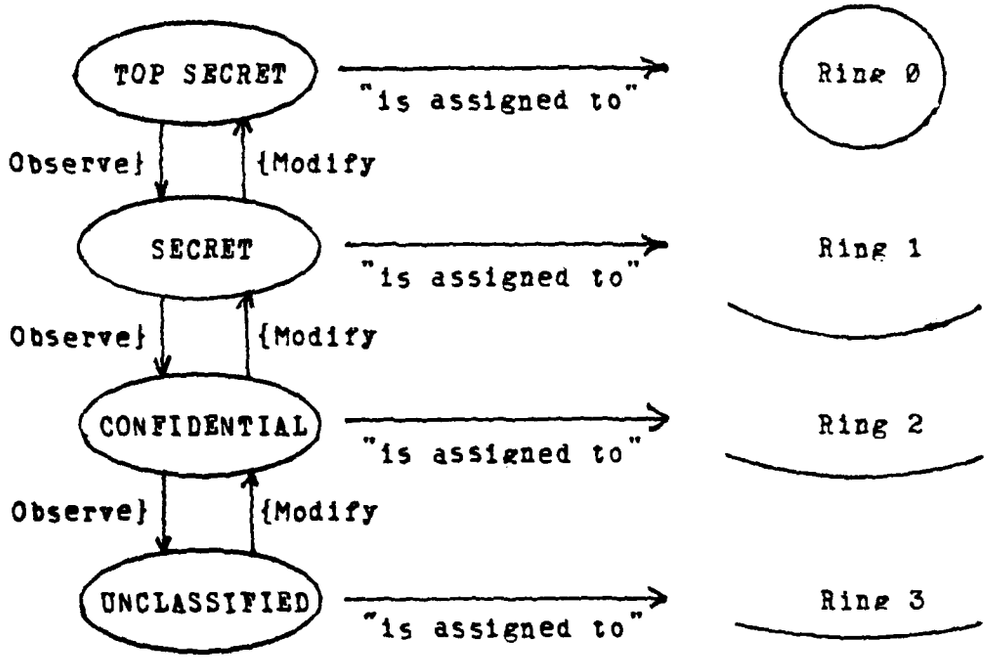


Figure 17. Basic National Security Assignment 1.

The Multics Ring Mechanism determines the authorized access of a process by means of the current ring number (r). Thus a process which is executing in ring number 1 would need to be cleared for at least SECRET information according to this assignment scheme.

The Multics Ring Mechanism discriminates among objects by means of a ring bracket. The ring bracket is a three-tuple (R1, R2, R3) where R1, R2 and R3 are ring numbers and $R1 \leq R2 \leq R3$. Access to objects is restricted such that the current ring of execution must be less than or

equal to R2 to observe information and less than or equal to R1 to modify information. Figure 18 shows characteristics of the ring brackets both in terms of the access modes used in this discussion and the access modes used in Multics.

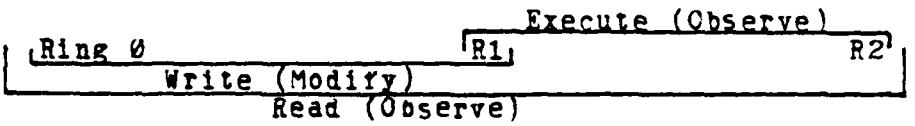


Figure 18. Multics Ring Mechanism.

Continuing now with the examination of access relations, consider an object that is classified as SECRET. Such an object must be assigned a ring bracket such that it may be observed by processes in ring 0 and ring 1 only. R2 must therefore be 1. This presents a problem. No matter what value one may choose for R1, a contradiction occurs. If R1 is 0 or 1 then TOP SECRET processes may modify SECRET files violating the Confinement Property. If R1 is greater than 1, the restrictions of the ring mechanism would be violated (viz., R1 > R2). Therefore, one can conclude that this assignment is not acceptable.

Consider now the only other potential assignment scheme where the greatest lower bound of the lattice (the UNCLASSIFIED access class) is assigned to ring 0. This assignment is illustrated in figure 19.

One may now attempt to assign ring brackets to an object classified SECRET. A problem occurs immediately. One

wants processes executing in ring 2 to observe SECRET objects, but then a process in ring 0 (i.e., an UNCLASSIFIED process), will also be able to observe the object. The Simple Security Condition cannot be enforced with this assignment, so the assignment scheme is not feasible.

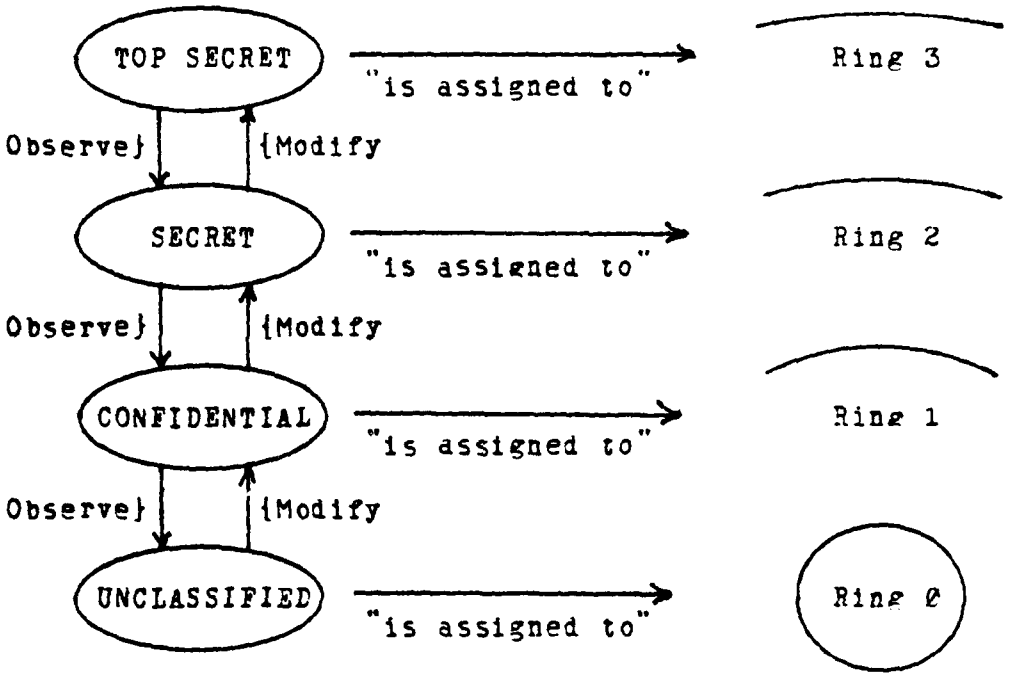


Figure 19. Basic National Security Assignment 2.

Since neither of these assignments are acceptable, and shifting the ring assignments numerically would yield similar results, one can see that no assignment will be acceptable. Therefore, the Multics Ring Mechanism is not sufficient to enforce the basic National Security policy for compromise.

2. Subversion Policy

The basic National Integrity policy [21] is the dual of the basic National Security policy. Whereas the security policy is concerned with the unauthorized observation of information or compromise, the integrity policy is concerned with the unauthorized modification of information or subversion as discussed in section II.

Consider first the assignment of the TOP SECRET access class (the least upper bound for the lattice established by the policy) to Ring 0 (the least upper bound for the dominance domains established by the mechanism). The assignment produced is shown in figure 20.

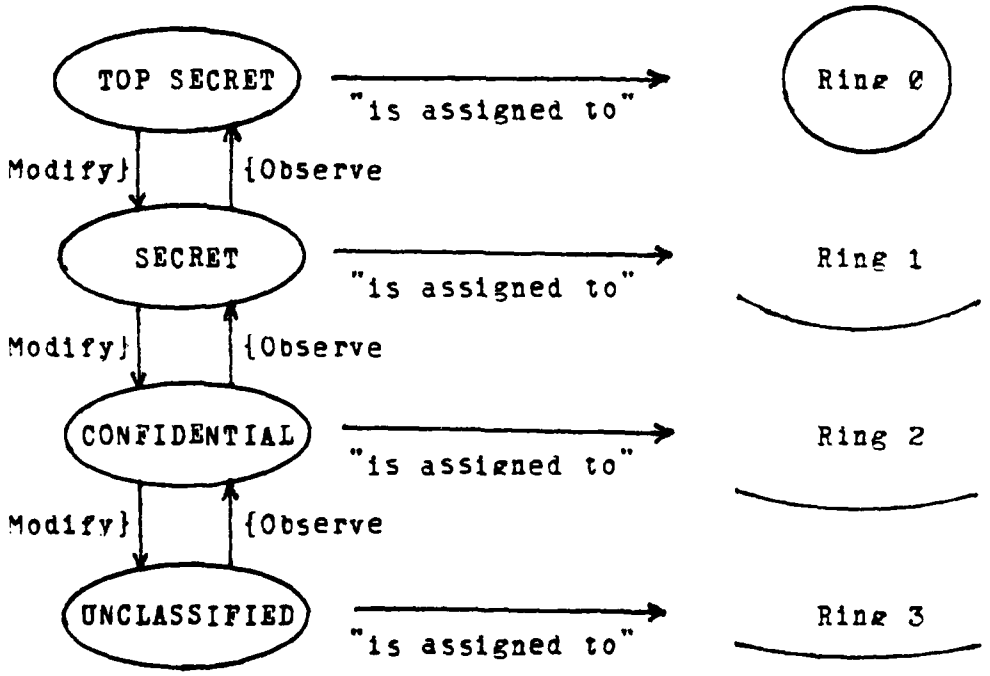


Figure 20. Basic National Integrity Assignment 1.

One may now examine the access relations which the Multics Ring Mechanism will permit (as shown in figure 18) and test for possible violations of the policy. In so doing, one encounters violations almost immediently. One wishes to have a process executing in Ring 1 (i.e., a SECRET process), for example, to be able to observe TOP SECRET objects in Ring Ø, but the mechanism prohibits this observation. Additionally, a SECRET process could observe CONFIDENTIAL information violating the Integrity Confinement Property. Therefore, this assignment scheme is not feasible.

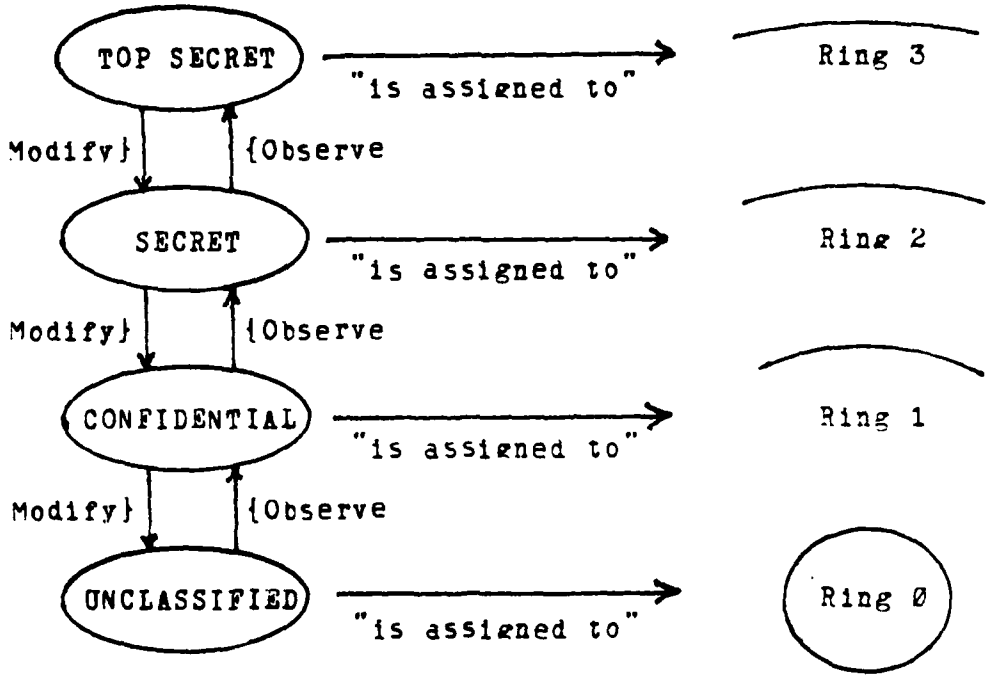


Figure 21. Basic National Integrity Assignment 2.

Consider now the only other potential assignment scheme (viz., according to the Simple Assignment Theorem)

where the TOP SECRET equivalence class is assigned to Ring 3. This assignment scheme is illustrated in figure 21.

Examining this assignment, consider an object that is classified as SECRET. Such an object must be assigned a ring bracket such that it may be observed by processes in Ring 0, Ring 1 and Ring 2 only, so R2 must be assigned 2. But if R2 is 2, one is faced with a contradiction in the assignment of R1. If R1 is assigned 0, 1 or 2, then a violation of the Simple Integrity Condition occurs because UNCLASSIFIED subjects may then modify SECRET objects. If R1 is assigned 3, the Ring Bracket constraints are violated. Therefore, this assignment scheme fails to provide an assignment where the protection mechanism can enforce this policy.

According to the Simple Assignment Theorem, there are no other assignments worthy of consideration. Therefore, the Multics Ring Mechanism is not sufficient to enforce this policy either.

So far, it has been shown that the Multics Ring Mechanism is not sufficient to enforce the basic National Security policy nor the basic National Integrity policy. However, a Multics Security Kernel has been designed [28,29] that is sufficient to support both of these policies. This may seem to be a contradiction but it is not. The confusion is dissipated when one asks the question, "What form of policy does the Multics Ring Mechanism support?"

3. Program Integrity Policy

The general form of Program Integrity policies was introduced in section II. Consider now the specific program integrity policy shown in figure 22.

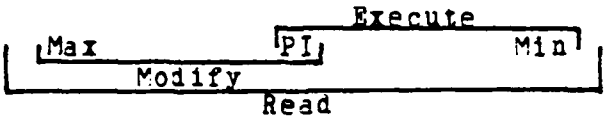


Figure 22. A Program Integrity Policy.

According to this policy, entities are partitioned into one of four access classes designated as User, Supervisor, Utility or Kernel. The sensitivity of these access classes is specified as : Kernel > Supervisor > Utility > User. An assignment to a Multics ring structure is made as shown in figure 23.

Recalling the characteristics of ring brackets shown in figure 18, "Max" is designated as Ring 0, the program integrity access class (PI) as R1 and "Min" as R2. One may note that for this policy any choice for R2 greater than or equal to R1 will do. This analysis, however, has fixed R2 at 3.

According to the assignment technique, one must now examine the access relations permitted by the mechanism and test for possible violations of the policy. Unlike previous examples, where the mechanism was obviously not sufficient to support the policy (i.e., only a single counter-example

was necessary) this example examines a policy that is likely to be supported by the Multics Ring Mechanism. Knowing this, it seems appropriate to present a more careful approach for the validation of this assignment.

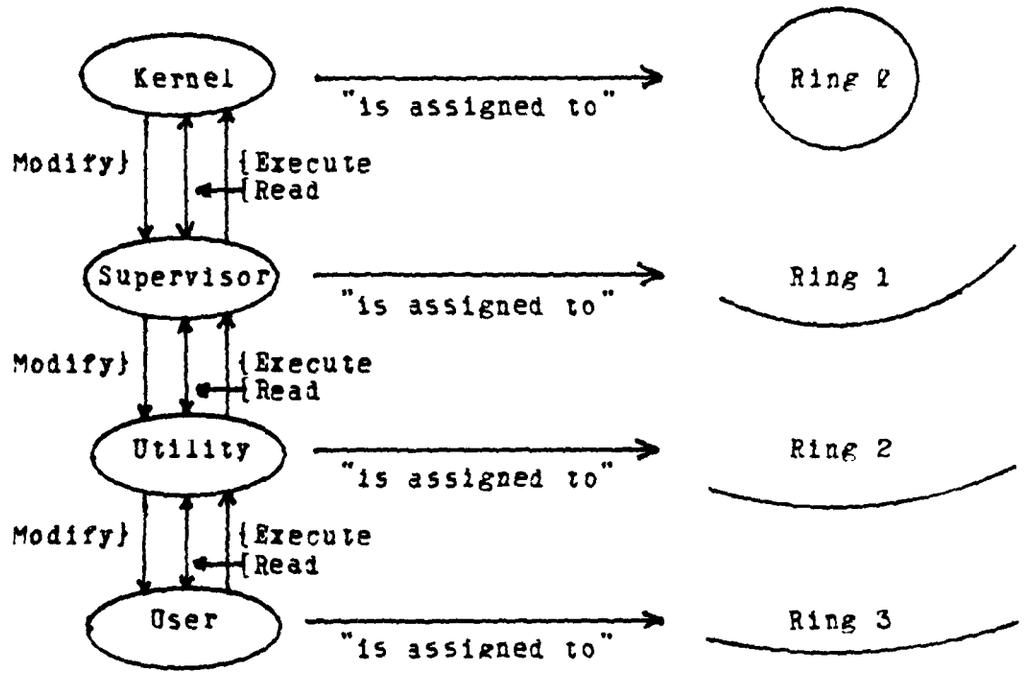


Figure 23. Program Integrity Assignment 1.

For simplicity, one may refer to e_0 (the first equivalence class) as Kernel (i.e., the access class that labels this equivalence class of subjects and objects), e_1 as Supervisor, e_2 as Utility and e_3 as User. One may also refer to Δ_0 (the first dominance domain established by the Multics Ring Mechanism) as Ring 0, Δ_1 as Ring 1, Δ_2 as Ring 2 and Δ_3 as Ring 3. The assignment scheme consists of

assigning e_0 to Δ_0 (Kernel to Ring 0), e_1 to Δ_1 (Supervisor to Ring 1), e_2 to Δ_2 (Utility to Ring 2), e_3 to Δ_3 (User to Ring 3). One can now evaluate the access relations permitted by the Multics Ring Mechanism and compare them with the policy.

Examining the read access first, one notes that the Multics Ring Mechanism provides no discrimination for read access since R2 is fixed at 3 for all objects. Thus subjects in Δ_0, Δ_1, Δ_2 or Δ_3 may read objects in Δ_0, Δ_1, Δ_2 and Δ_3. This corresponds with the access rights of the policy which states that subjects in e_0, e_1, e_2 or e_3 may read objects in e_0, e_1, e_2 and e_3. Therefore, the mechanism is sufficient with respect to the read access relations.

Next, examining the modify access relations one may observe that $M\Delta_0 \propto M\Delta_1 \propto M\Delta_2 \propto M\Delta_3$. Thus a subject in Δ_0 may modify objects in Δ_0, Δ_1, Δ_2 or Δ_3. This corresponds to the access rights of the Kernel access class in that a subject in e_0 may modify objects in e_0, e_1, e_2 and e_3. Examining Δ_1, one observes that a subject in Δ_1 may modify objects in Δ_1, Δ_2 or Δ_3 but not in Δ_0. This corresponds with the access rights of the Supervisor access class in that a subject in e_1 may modify objects in e_1, e_2 and e_3 but not in e_0. Examining Δ_2, one observes that a subject in Δ_2 may modify objects in Δ_2 or Δ_3 but not in Δ_0 or Δ_1. This corresponds with the access rights of the Utility access class in that a subject in e_2 may modify

objects in e_2 or e_3 but not in e_0 or e_1. Finally, examining Δ_3, one observes that a subject in Δ_3 may only modify objects in Δ_3. This corresponds with the access rights of the User access class in that a subject in e_3 may only modify objects in e_3. Therefore, the Multics Ring Mechanism is sufficient to support this policy with respect to modify access relations.

Next, examining the execute access relations one may observe that $X\Delta_3 \prec X\Delta_2 \prec X\Delta_1 \prec X\Delta_0$. This is just the inverse of the modify access relations. Thus a subject in Δ_3 may execute objects in Δ_0, Δ_1, Δ_2 or Δ_3. This corresponds to the access rights of the User access class in that a subject in e_3 may execute objects in e_0, e_1, e_2 and e_3. Examining Δ_2, one observes that a subject in Δ_2 may execute objects in Δ_0, Δ_1 or Δ_2 but not in Δ_3. This corresponds with the access rights of the Utility access class in that a subject in e_2 may execute objects in e_0, e_1 and e_2 but not in e_3. Examining Δ_1, one observes that a subject in Δ_1 may execute objects in Δ_0 or Δ_1 but not in Δ_2 or Δ_3. This corresponds with the access rights of the Supervisor access class in that a subject in e_1 may execute objects in e_0 or e_1 but not in e_2 or e_3. Finally, examining Δ_0, one observes that a subject in Δ_0 may only execute objects in Δ_0. This corresponds with the access rights of the Kernel access class in that a subject in e_0 may only execute objects in e_0. Therefore, the

Multics Ring Mechanism is sufficient to support this policy with respect to execute access relations.

So one may observe that for each of the access modes (read, modify and execute), the Multics Ring Mechanism is sufficient to enforce the policy. Therefore, for this assignment, no violations are possible, thus proving that the Multics Ring Mechanism is sufficient to support this Program Integrity policy.

B. OTHER RING MECHANISMS

The Multics Ring Mechanism is by no means the only form of Ring Mechanism. By altering the requirements of the Ring Brackets and the need for a Gate Keeper, one can contemplate adapting the ring mechanisms to meet other simple hierarchical policies.

Consider using the assignment shown in figure 17, but altering the means of discrimination among objects such that the Ring Bracket is a singleton (R1). Following the rules shown in figure 24, one can adapt this ring mechanism to enforce the basic National Security policy.

Figure 24. Security Rings.

Similary, figure 25 shows the rules necessary for the same assignment as shown in figure 20 to adapt this ring mechanism to meet the basic National Integrity policy.

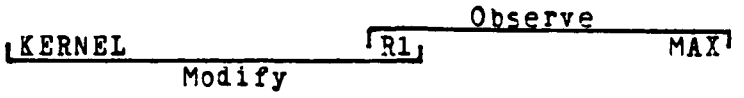


Figure 25. Integrity Rings.

To be sure, these brief suggestions do not completely characterize a practical protection mechanism. However, it appears that ring mechanisms are adaptable for the enforcement of various simple hierarchical policies.

C. CAPABILITY MECHANISMS

Considerable effort is currently underway to provide "Provably Secure Operating System" based upon the capability mechanism [30,31]. It is important to examine what form of protection capabilities actually provide.

Capability mechanisms primarily establish two dominance domains that are enforced by this system hardware mechanism. One domain consists of capabilities, and the other is objects that are not capabilities such as segments and directories. A process takes no note of these dominance domains, however, because all processes have access to capabilities as well as other types of objects. So with respect to a process, the capability mechanism provides no inherent partitioning of the system entities at all. In

fact, in trying to determine the structure of dominance domains for non-capability objects, one encounters a veritable "spaghetti bowl" of domains, devoid of any inherent, unifying structure. Thus a capability mechanism is of itself not sufficient for the enforcement of any non-discretionary security policy. Enforcement of non-discretionary security policies (i.e., those of primary interest to National Defense) must be accomplished by some other add-on mechanism.

This is not to say that a capability mechanism is not useful. For example, the mechanism can protect a security kernel in much the same way as rings protect the kernel in the Multics design.

The usefulness of the assignment technique in validating the suitability of a protection mechanism to enforce a security policy has been examined in this section. The validity of the assignment technique has been established.

VI. CONCLUSION

This research has explored the foundations of non-discretionary security, discovering an effective methodology for assessing the sufficiency of a protection mechanism to enforce a non-discretionary security policy. By formalizing the notion of a domain [6,7], and using a formal notion of non-discretionary security [3], the inseparable nature of protection mechanisms and security policies has been established. This section considers some future directions for research and summarizes the principle findings of the author.

A. FUTURE DIRECTIONS

Although this author's investigation has provided some structure to the complex nature of security, considerable research is still needed. The relationship between protection mechanisms and other operating systems mechanisms is not clear. Such issues as serializability, synchronization and distributed processing may add new dimensions to the meaning of protection. Fundamental limitations regarding implementation details remain unknown.

Additionally, one can consider the formalization of policy specifications in general. Can the enforcement of any policies other than lattice policies be evaluated? Can all

enforceable policies be represented in some common form such as a lattice?

One of the most difficult problems in actually enforcing any security policy is the maintenance of unique non-forgeable attributes [6] associated with the subjects and objects. A mechanism for maintaining the uniqueness of these attributes may be called an "isolation mechanism" because it isolates those subjects that may access these attributes from those that may not. This does not prevent sharing of objects but simply provides a means of isolating these attributes from general unprotected usage. Both the capability mechanism [30,31] and the notion of a gate (mechanism) [8,28] appear to be isolation mechanisms. A comprehensive study of this problem is beyond the scope of this discussion. However, a few observations concerning isolation noted during this research are provided.

The fundamental principles upon which an isolation mechanism must rely is the notion of a segment (i.e., an atomic unit of information storage for which the access class is identified) and the tranquillity principle (i.e., the notion that the access class for a subject or an object does not change during the course of computations) [17]. If these two principles are not enforced, it is not clear how one may evaluate the enforcement of any non-discretionary security policy.

The tranquillity principle does not strictly apply to processes. In Multics, for example, processes had several domains of execution. However, since a subject is defined as a process-domain pair, one might at first suspect that a process executing in multiple domains does not present a security problem. This is not always the case, particularly when dealing with policies that attempt to limit the information flow [13].

When attempting to enforce the National Security Policy in a multi-user, multi-process environment, where a process executes in a sequential fashion (i.e., the process is serializable) one can do no better than to allow a process to proceed to its "high water mark" and then terminate at that level. Any attempt to revert to a less sensitive access class will result in a potential compromise. For example, consider the compromise technique shown in figure 26.

In this example, a malicious agent utilizes the feature of sequential processes and the basic PV synchronization mechanism [33] to take the "Info" in Dominance Domain 2 and copy it into Dominance Domain 1. In order to do so, the agent calls procedures placed in the "High" domain by subversion [3], relying only upon one process (i.e., PROCESS 0 or PROCESS 1) to return, thus providing the information in binary form to the "Low" domain. Thus by serialization and process synchronization alone, the isolation of the dominance domains has been compromised.

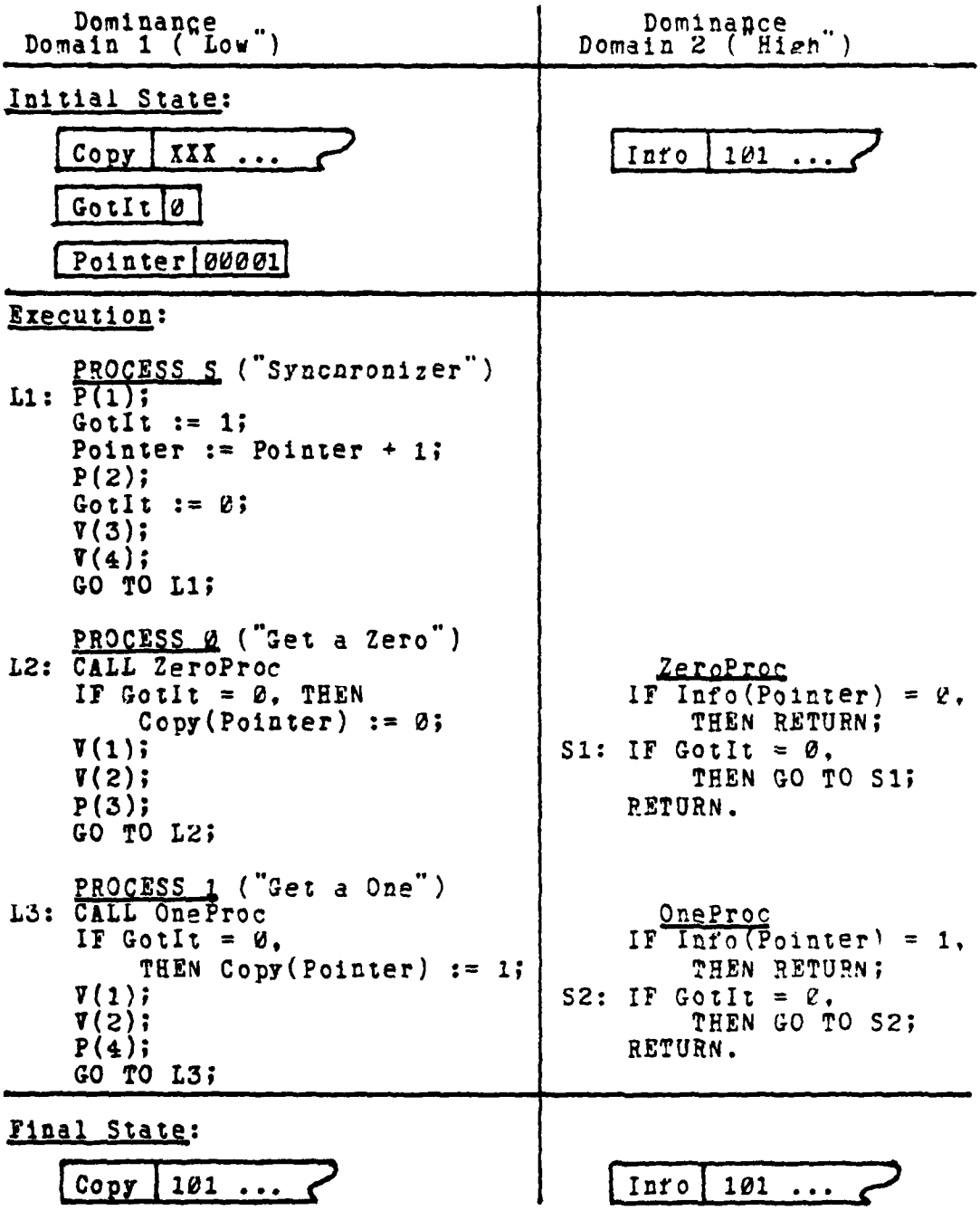


Figure 26. Serialization Problem.

Note that were the processes to act independently in each dominance domain (i.e., processes are serializable only with respect to a given dominance domain or synchronization between two processes is not possible) this compromise could not occur. In general, this example shows that synchronization of processes, serialization of processes and secure computations are fundamentally related in some fashion. The exact nature of this relationship is not clear.

B. RESULTS

The assignment technique has been shown to be a useful method for validating the sufficiency of a protection mechanism to enforce non-discretionary security policies. This method provides considerable insight into the nature of access control. One may observe that non-discretionary security is dependent only upon the dominance domains established by the systems mechanisms and their associated permissible access relations. The nature of the computation is of no concern.

Any non-discretionary security policy for which the access classes and access relations can be enumerated, can be enforced in a theoretical sense. Actual implementation, however, is dependent upon the systems' isolation mechanism. No policy can be enforced, in a practical sense, unless the system can maintain unique non-forgeable attributes.

Protection mechanisms inherently "mirror" the policies that they enforce. Non-discretionary Security policies form a lattice of access classes that may be mapped to an isomorphic image of dominance domains, inherently established by the protection mechanism. Since this has been shown, one need not illustrate separate lattices for both policy and mechanism. One unified description for both the lattice policy and its image established by the protection mechanism is sufficient for general systems design considerations.

One may also consider approaching the assignment technique from the mechanism point of view. The question then becomes, "Given some general Protection Mechanism, what form of policies will it support?" An absolute answer to this question is, in general, not available. However, one can make an evaluation for those policies that are of current interest. Thus, the assignment technique gives one a forum in which to consider the usefulness of protection mechanisms for specific policies of interest.

"Uniform protection mechanisms," i.e., those mechanisms forming lattice structures of dominance domains where the access relations between any two antisymmetric dominance domains are identical, may be represented by linear access graphs in the same manner as a policy. When the linear access graph for the policy is similar to the linear access graph for the mechanism, one can see that for a carefully

chosen assignment scheme, the protection mechanism will enforce the security policy.

One may consider the development of a taxonomy of uniform protection mechanisms based upon the nature of the access control that each enforces. Such a taxonomy is beyond the scope of this discussion, however, the linear access graphs illustrated throughout this text may be helpful in initiating such an effort.

The protection provided by the Multics Ring Mechanism appears to be precisely the issue that Wulf, Jones and the other designers of the "HYDRA" system were attempting to understand [18]. They introduce their discussion by first saying :

> "Protection is, in our view, a mechanism." [18]

Their discussion then proceeds to make the following general statement relative to the Multics rings:

> "Our rejection of hierarchical system structures and especially ones which employ a single hierarchical relation for all aspects of system interaction, is also, in part, a consequence of the distinction between protection and security. A failure to distinguish these issues coupled with a strict hierarchical structure leads inevitably to a succession of increasingly privileged system components, and ultimately to a "most privileged" one, which gain their privilege exclusively by virtue of their position in the hierarchy. Such structures are <u>inherently wrong</u> ..." [18]

Had the assignment technique been available to the authors of the above statement, they would have been afforded a means of expressing their views more precisely than the ambiguous phrase "inherently wrong". The assignment technique provides a precise means for clearly formulating such an observation and evaluating its validity. As shown in section V, and in agreement with Wulfs' statement, the Multics Ring Mechanism is "inherently wrong" with respect to compromise policies. On the other hand, the Multics Ring Mechanism is "just right" as a means of enforcing a program integrity policy or assisting in the enforcement of the systems hierarchical as well as non-hierarchical security policies (viz., via Security Kernels).

Additionally, in the same report [18] the authors make the following observation with respect to their overall design methodology :

> "Among the major causes of our inability to experiment with, and adapt, existing operating systems is their failure to properly separate mechanisms from policy." [18]

The assignment technique has shown, however, that lattice security policies and protection mechanisms that enforce these policies are inextricably related. Recognizing this inseparability should provide considerable insight into current efforts in this area.

Overall, assignment research has provided a mathematical methodology for unifying the discussion of security related issues. One may now properly refer to an access mode as a realization of an access right, a dominance domain as a realization of an access class and a protection mechanism as a realization of a security policy.

LIST OF REFERENCES

1. Schell, R. R., "Computer Security: The Achilles' Heel of the Electronic Air Force?", Air University Review, Jan-Feb 1979, p. 16-32.

2. Myers, P. A., Subversion: The Neglected Aspect of Computer Security, Masters Thesis, Naval Postgraduate School, June 1980.

3. Schell, R. R., "Security Kernels: A Methodical Design of System Security," USE Technical papers (Spring Conference, 1979), March 1979, p. 245-250.

4. Shirley, L. J. and Schell, R. R., "Mechanism Sufficiency Validation by Assignment," Proceedings of Second IEEE Symposium on Privacy and Security, in preparation.

5. Dennis, J. B. and Van Horn, E. C., "Programming Semantics for Multiprogrammed Computations," Communications of the ACM, March 1966, p. 143-155.

6. Lampson, B. W., "Dynamic Protection Structures", AFIPS Conf. Proc. 35, FJCC 1969, p. 27-38.

7. Lampson, B. W., "Protection", Proceedings Fifth Annual Conference on Information Sciences and Systems, Princeton University, March 1971, p. 437-443.

8. Schroeder, M. D., Cooperation of Mutually Suspicious Subsystems in a Computer Utility, (AD-750 173), Ph. D. Thesis, Massachusetts Institute of Technology, September 1972.

9. Popek, G. J., Access Control Models, Harvard University Report, ESD-TR-73-106 (AD-761 907), February 1973.

10. Bell, D. E. et. al., Secure Computer Systems: Mathematical Foundations, MITRE Corporation Report, ESD-TR-73-278-VOL-1, (AD-770 768), November 1973.

11. Furtek, F. C., A Validation Technique for Computer Security Based on the Theory of Constraints, MITRE Corporation Report, ESD-TR-78-182, (AD-A065 111), December 1978.

12. Walter, K. G. et. al., Primitive Models for Computer Security, Case Western Reserve University Report, ESD-TR-74-117, (AD-778 467), January 1974.

13. Denning, D. E., "A Lattice Model of Secure Information Flow," Communications of the ACM, May 1976, p. 236-243.

14. Saltzer, J. H., and Schroeder, M. D., "The Protection of Information in Computer Systems", Proceedings IEEE, September 1975, p. 1278-1308.

15. Cohen, E. S., Problems, Mechanisms and Solutions, Ph. D. Thesis, Carnegie-Mellon University, AFOSR-TR-77-00006, (AD-A034 855), August 1976.

16. Gronn, M. J., A Model of a Protected Data Management System, Canadian Commercial Corporation Report, ESD-TR-76-289, (AD-A035 256), June 1976.

17. Bell, D. E. and LaPadula, L. J., Secure Computer System: Unified Exposition and Multics Implementation, MITRE Corporation Report ESD-TR-75-306, (AD-A023 588), March 1976.

18. Wulf W. et. al., Hydra: The Kernel of a Multiprocessor Operating System, Carnegie-Mellon University Report, AFOSR-TR-73-1079, (AD-762 514), June 1973.

19. Jones, A. K., "Protection Mechanism Models: Their Usefullness", in Foundations of Secure Computations, by R. A. DeMillo et. al., New York: Academic Press, 1978, p. 237-254.

20. Snyder, L., "Formal Methods of Capability-Based Protection Systems", IEEE Transactions on Computers, March 1981, p. 172-181.

21. Biba, K. J., Integrity Considerations for Secure Computer Systems, MITRE Corporation Report, ESD-TR-76-372, (AD-A039 324), April 1977.

22. Stanat, D. F., and McAllister, D. F., Discrete Mathematics in Computer Science, Prentice Hall Inc., 1977.

23. Gericke, H., Lattice Theory, Frederick Ungar Co., 1966.

24. Jones, A. K., Protection in Programmed Systems, Ph. D. Thesis, Carnegie Mellon University, 1973.

25. Office of the President of the United States, Executive Order 12065, 26 June 1978.

26. Turn, R. and Ware W. H., "Privacy and Security in Computer Science," American Scientist, 1975.

27. Schroeder, M. D. and Saltzer, J. H., "A Hardware Architecture for Implementing Protection Rings," Communications of the ACM, March 1972, p. 157-170.

28. Schroeder, M. D. Clark, D. D., and Saltzer, J. H., "The Multics Kernel Design Project," Proceedings of Sixth ACM Symposium on Operating Systems, November 1977, p. 43-56.

29. Whitmore, J. et. al., Design for Multics Security Enhancements, Honeywell Information Systems Report, ESD-TR-74-176, (AD-A030 801), December 1973.

30. Feirtag, R. J. and Neuman, P. G., "The Foundations of a Provably Secure Operating System (PSOS)", AFIPS National Computer Conference Conference, 1979, p. 329-334.

31. Neumann, P. G., Robinson, L., Levitt, K. N., A Provably Secure Operating System, Stanford Research Institute Report, (AD-A088 601), June 1975.

32. Department of Defense, DOD 5200.1R, DOD Information Security Program Requirements.

33. Dijstra, E.W., "The structure of "THE"-Multiprogramming System," Communications of the ACM, May 1968, p. 341-346.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Post Graduate School Monterey, California 93940	2
3.	Office of Research Administration Code 012A Naval Post Graduate School Monterey, California 93940	1
4.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93940	2
5.	COL Roger R. Schell 6940th ESW Box 2999 Fort George G. Meade, Md. 20755	5
6.	Lyle A. Cox, Jr., Code 52Cl Department of Computer Science Naval Postgraduate School Monterey, California 93940	4
7.	Commandant, U. S. Coast Guard (G-FIS) 2100 2nd St. S.W. Washington, D.C. 20590	2
8.	Commandant, U. S. Coast Guard (G-PTE) 2100 2nd St. S.W. Washington, D.C. 20590	2
9.	Lawrence J. Shirley Commandant, U. S. Coast Guard (G-FIS) 2100 2nd St. S.W. Washington, D.C. 20590	10

10. James P. Anderson 1
Box 42
Fort Washington, Pa. 19034

11. Terry S. Arnold 1
2333 Camino Del Rio South
Suite 250
P. O. Box 20217
San Diego, Ca. 92120

12. J. W. Freeman 1
Computer Sciences Corp.
803 West Broad Street
Falls Church, Virginia 22046
Attn: Ingrid Mues

13. Bruce Goldstein, Pres. 1
Executect
1100 Gough Street, Suite 8F
San Francisco, Ca. 94109

14. Daniel G. Roblyer 1
The Aerospace Corporation
P. O. Box 92957
Los Angles, Ca. 90009

15. Marvin Schaefer 1
System Development Corporation
2500 Colorado Avenue
Santa Monica, Ca. 90406

16. John P. Schill 1
Code 8321 Concept Development Branch
C3 I Sys Dept
Naval Ocean Systems Center
San Diego, Ca. 92152

17. Peter S. Tasker 1
The Mitre Corporation
P. O. Box 208 MS/B-332
Bedford, Mass. 01730

18. Rein Turn 1
California State University, Northridge
Department of Computer Science
18111 Nordhoff Street
Northridge, Ca. 91330

19. Kyle E. White 1
P. O. Box 1821
Vandenburg AFB, Ca. 93437

20. John Woodward 1
The Mitre Corporation
P. O. Box 208
Bedford, Mass. 01730

21. Kathryn Heniger, Code 7503 1
Naval Research Lab
Washington, D. C. 20375

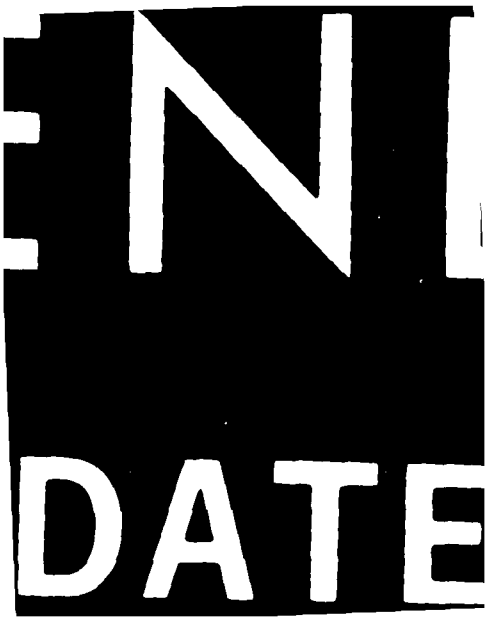
22. Joel Trimble, Code 221 1
Office of Naval Research
800 North Quincy
Arlington, Va. 22217

23. Carl Landwehr 1
Naval Research Laboratory Code 7593
Washington, D. C. 20375

24. Steven B. Lipner 1
Digital Equipment Corp.
ML3-2/E41
146 Main Street
Maynard, Mass. 01754

25. Prof. Tien. Tao, Code 62TV 1
Department of Computer Science
Naval Postgraduate School
Monterey, California 93940

26. Lcdr. W. Snockley, Code 52SB 1
Department of Computer Science
Naval Postgraduate School
Monterey, California 93940


N
DATE